AFRL-SR-AR-TR-04-

0409

REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
	Final	15 June 1997 - 14 December 2002

4. TITLE AND SUBTITLE
Building and Acquiring Interactionist Ontologies

5a. CONTRACT NUMBER

5b. GRANT NUMBER
F49620-97-1-0485

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Paul R. Cohen

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computer Science Department
University of Massachusetts
Amherst, MA 01003

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
4015 Wilson Blvd
Mail Room 713
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Our HPKB effort was intended to determine whether physical schemas could be a semantic core for ontologies. This report describes the basic theory of physical schema and our work on re-use of knowledge from ontologies. An addendum to the report is a published summary of all the research in the entire HPKB program written by Prof. Cohen (first author) and many of the HPKB principals.

15. SUBJECT TERMS

20040810 021

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Paul R. Cohen
					19b. TELEPHONE NUMBER *(Include area code)* 413-545-3638

8-04-04

NM

Building and Acquiring Interactionist Ontologies

Final Technical Report

Principal Investigator: Paul R. Cohen
Experimental Knowledge Systems Laboratory
Computer Science Department
University of Massachusetts
Amherst, MA 01003

cohen@cs.umass.edu
413-545-3638

Sponsored by the
Air Force Office of Scientific Research
Contract Number: F49620-97-1-0485

Program Manager: Dave Gunning

Effective Date of Contract: June 15, 1997
Contract Expiration Date: December 14, 2002
Contract amount: $995,130.57

Technical Final Report

Our HPKB effort was intended to determine whether *physical schemas* could be a *semantic core* for ontologies. This report describes the basic theory of physical schemas and our work on re-use of knowledge from ontologies. An addendum to the report is a published summary of all the research in the entire HPKB program written by Prof. Cohen (first author) and many of the HPKB principals.

Physical Schemas and the Semantic Core

Many situations can be understood in terms of relatively few, physical relationships called image schemas or physical schemas (Lakoff 1984, Lakoff and Johnson 1980). Let's consider some examples:

In/Out The first preposition learned by children of all cultures is "in." This is because "in" corresponds to one of the most basic *conceptual* structures in our minds, the idea of inclusion, containment, being part of a group or activity, being in a state, and so on. If an intelligent agent understood the word "in" as we do, then it would understand that the following sentences share a core "in" meaning:

The base is in the Changjong Sector.
The attack is in progress.
The red forces are contained.
The wing is deep in enemy airspace.
You're in the army, now.
The troops are in a confident state of mind.

Similarly, a core "out" meaning is common to the following sentences:

The base is outside the theater of operations.
The attack is not our problem.

In one of these sentences "outside" has a spatial interpretation; the other sentence evokes a mental space of things we are concerned with, and says the attack isn't in that space. Whether we are referring to political boundaries, military objectives, organizations, or states of mind, there is a common notion of "in" and "out" in all these sentences: Objects or activities are "within" or conversely "outside" physical, mental, or intentional boundaries.

Force and Resistance. Long before they learn any words at all, children learn that their actions have effects. This is the notion of imposing one's will on the physical and mental world, what we call a core "force" meaning:

The forces have been made to retreat.
Bill convinced John to change his mind.
We pushed them back across the river.
We took the Changjong Sector.
We evacuated the area.

The core "force" meaning is fundamentally physical, although it is sometimes extended to forcing a change in mental state or abstract states like ownership. Of course, one cannot always impose one's will on the world. It sometimes kicks back, or simply impedes:

We can't get through the AA defenses.
We are unable to cross the river.
We are meeting stiff resistance.
John won't change his mind, despite Bill's entreaties.

The core "resistance" meaning is also fundamentally physical; it evokes a mental image of pushing against something that won't yield, or yields slowly, or pushes back.

Control The concept of control is closely related to force and resistance, but it implies ongoing activities for an indefinite period. Whereas force is applied to a steering wheel, which offers resistance, "steering" refers to the ongoing process of controlling the car by application of force to the steering wheel:

We take our orders from General Smith.
We control the Changjong Sector.
They are our prisoners (i.e., we control them).

Path As a final example, consider the abstract schema of starting *here* and ending *there*, which we call the "path" image schema:

We drove to the Changjong Sector.
The orders have come down.
The operation is going according to plan.
The air defenses are being repaired.

The first of these sentences is a straightforward, physical "path" meaning: physical agents went from one physical location to another. The second sentence extends the "path" meaning to transferring information. In the third and fourth sentences, the path isn't physical, but rather, is a plan or activity that's being followed.

Conceptual structures like in/out, force/resistance, control and path are called image schemas in the literature, although the name is a bit confusing because these structures are not images in the sense of being mental pictures. They are more basic than images,

and independent of sensory modalities. The claim, borne out in the psychological literature, is that it doesn't matter whether you see, feel, or hear of a force/resistance situation, the same basic structure is evoked in your mind, and the evocation of this structure is essential to your understanding of the situation. This point, that image schemas are central to understanding situations, is discussed in detail, below.

Intelligent agents for command and control activities will be able to understand situations as humans do---and thus, will be most helpful to commanders---if they have similar mental models. Humans conceptualize campaigns in particular ways, and we intend to build intelligent agents that conceptualize campaigns in similar ways. The conceptualization of a campaign will be called a *campaign model.* If intelligent agents and humans have congruent campaign models, then they will understand the campaign in the same way, specifically, agents will detect the same opportunities and pitfalls, and draw the same inferences as humans, and communication between humans and agents will be more efficient.

Our principal hypothesis was that if campaign models are constructed from image-schematic components such as in/out, force/resistance, path, then the models will be congruent with those of human campaign planners. Here are some related claims: Image-schematic campaign models are the basis for integrating visualization, simulation, planning and other command and control activities into a single intelligent assistant. We can provide mental models that will enable human planners to rapidly configure simulations and visualizations for specific campaigns. Image-schematic campaign models provide a semantics for inferences about the future; they enable an automated system to say not only what may happen, but also what it means in terms of the goals of the campaign plan.

Evidence for the Hypotheses

Three kinds of evidence is offered for the ubiquity of image schematic semantic primitives. The first is the Capture the Flag wargaming simulator, which is based entirely in image schemas. The code to control agents in the simulator is written in terms of primitives like *path, apply force, containment*, and so on. The fact that we could build the simulator and behaviors for agents is prima facia evidence that these primitives are sufficient for human experts to express their knowledge about physical actions of units on the battlefield.

The second kind of evidence is from a study of human verbs. We present an image-schematic theory of verb semantics called the Maps for Verbs framework.

The third kind of evidence is indirect and doesn't speak to image schemas per se but does show that primitive ontological knowledge is re-used by knowledge engineers. This means the effort of working out a core semantics is apt to be rewarded.

We present these arguments in turn.

Capture the Flag

At the University of Massachusetts we have developed a simulation of war games and a planner, called Capture the Flag, which beats human adversaries in more than half the games it plays. Red forces are controlled by the planner, Blue by a human adversary. Some features of the game are roughly realistic: mobility is influenced by terrain, attrition is modeled by Lanchester equations, and Red forces are coordinated by tactics. Most importantly, tempo strongly influences outcomes. When Blue loses the tempo, Red presses its advantage. A tactical disadvantage quickly spreads to a scenario-wide loss of initiative, Blue becomes reactive, and eventually loses the game.

Although the planner currently plays autonomously, it is intended as a mixed-initiative assistant to human planners. A fielded planner would support course of action development, helping commanders to evaluate alternatives and to "think outside the box." In war games the planner can play autonomously as an intelligent adversary, or as an assistant. Capture the Flag is intended to have the same pedagogical roles as strong chess algorithms: providing a fast, unflagging, powerful opponent for students of tactics and strategy.

The planner wins because it considers a huge number of tactical combinations, it continuously re-evaluates its commitments, it has a better sense of timing than human adversaries, and it doesn't lose track of its assets in complex situations, as humans do. These advantages are interrelated; in particular, timing and tactics are inextricable.

Physical Schema Planning

Capture the Flag is based on the idea that many physical processes are built from a small set of *physical schemas* such as *push, move, apply force, block, contain, follow*, and so on. Furthermore, these schemas are *primitive* in the sense that every child learns them very early in life, and uses them to plan his or her activities and interpret the activities of others. There is even some evidence that non-physical processes may be grounded in these simple, primitive physical schemas: Lakoff and Johnson, for example, make a strong case that much metaphor involves representing nonphysical things and processes as physical. For example, we speak of grasping an idea, containing information, pressing an advantage, facing an uphill battle, turning up the heat, and so on. I believe that physical schemas really are the foundation of much of what we know, that they explain how sensorimotor agents like infants make the transition to cognitive agents like us. I believe that planning a military campaign involves very similar reasoning as planning an offense in football, or a continuation in chess, or a path through heavy traffic. When my daughter pushes blocks around on her table and calls them cars, she is telling me that cars and her blocks have much in common – mass, velocity, rigid construction – and that the interactions between cars can be simulated by pushing blocks past each other, into each other, lining them up, and so on.

The language of military tactics is essentially physical, and has been at least since Clausewitz introduced physics into the study of warfare. Here are a few excerpts from Clausewitz, selected pretty much at random:

> "The conduct of war resembles the workings of an intricate machine with tremendous friction, so that combinations which are easily planned on paper can be executed only with great effort. Consequently the commander's free will and intelligence find themselves hampered at every turn, and remarkable strength of mind and spirit are needed to overcome this resistance."
> "... there is no higher and simpler law of strategy than that of *keeping one's forces concentrated.*"
> "... the stronger force not only destroys the weaker, but ... its impetus carries the weaker force along with it. ... In practice this is true, but only when war resembles a mechanical thrust."

Clausewitz relies on physical metaphors for his characterizations both of units on the battlefield and also for command and control. He views command as a kind of force, sufficient to overcome the friction encountered as a plan is executed.

Because we view tactical warfare as comprising physical processes such as movement, applying force, blocking, supporting, and so on, our Capture the Flag system has three parts:

The Abstract Force Simulator (AFS). Processes in AFS are modeled as interactions of masses, called *blobs*. Blobs have a small set of physical features, including mass, velocity, friction, radius, attack strength, and so on. A blob is an abstract unit; it could be an army, a soldier, or a political entity. Every blob has a small set of primitive actions it can perform, primarily move and apply-force. All other physical schemas are built from these primitives. Simply by changing the physics of the simulator, that is, how mass is affected by collisions, the friction for a blobs moving over types of surfaces, the resilience of units to collisions, and so on., we can transform AFS from a simulator of military units into a simulator of billiard balls.

AFS is a tick-based simulator, but the ticks are small enough to accurately model the physical interactions between blobs. Although blobs themselves move continuously in 2D space, for reasons of efficiency, the properties of this space, such as terrain attributes, are represented as a discrete grid of rectangular cells. Such a grid of cells is also used internally to bin spatially proximal blobs, making the time complexity of collision detection and blob sensor modeling no greater than linear in terms of the number of blobs in the simulator. AFS was designed from the outset to be able to simulate large numbers (on the order of hundreds or thousands) of blobs. The physics of the simulation are presently defined by the following parameters:

Blob-specific attributes:
- maximum acceleration and deceleration
- friction of the blob on different surfaces
- viscosity and elasticity: do blobs pass through one another or bounce off?

Global parameters:
- the effect of terrain on blobs

- the different types of blobs present in the simulation (such as blobs that need sustenance)
- the damage model: how blobs affect each others' masses by moving through each other or applying force.
- sustenance model: do blobs have to resupplied in order to prevent them from losing mass?

AFS allows us to express a blob's internal structure by composing it from smaller blobs, much like an army is composed of smaller organizational units and ultimately individual soldiers. But we don't have to take the internal structure into account when simulating, since at any level of abstraction, every blob is completely characterized by the physical attributes associated with it. Armies can move and apply force just like individual soldiers do. The physics of armies is different than the physics of soldiers, and the time and space scales are different, but the main idea behind AFS is that we can simulate at the "army" level if we so desire – if we believe it is unnecessary or inefficient to simulate in more detail.

Since AFS is basically just simulating physics, the top-level control loop of the simulator is quite straightforward: On each tick, loop over all blobs in the simulator and update each one based on the forces acting on it. If blobs interact, the physics of the world will specify what form their interaction will take. Then update the blob's low-level sensors, if it has any. Each blob is assumed to have a state reflector, a data structure that expresses the current state of the blob's sensory experience. It is the simulator's job to update this data structure.

Hierarchical Agent Control

The blob control architecture is hierarchical. We use the physical primitives *move* and *apply-force* to construct schematic plans for domain-specific actions like convoy and sneak-attack. Higher levels of control provide goals and context for the lower levels, and lower levels provide sensory reports, messages, and errors to the higher levels. A higher level cannot overrule the sensory information provided by a lower level, nor can a lower level interfere with the control of a higher level.

The AFS control architecture provides facilities for sensor management, action scheduling, message passing, and resource arbitration. Since all of AFS's actions are physically grounded, we can even control real-life robots.

The GRASP Planner

Capture the Flag has many agents and flags on each side. Any generative planning solution would face an enormous branching factor since many possible action combinations can be executed at any given time. To cope with this problem, we rely on a partial hierarchical planner, which retrieves plans from a set of pre-compiled skeletal plans, and uses heuristics to allocate resources in a reasonable way (for example, an *attack* plan will rarely attack a target with a smaller force than the force defending it).

When several plans apply, military planners will play out a plan and determine how the opponent might react to it. A wargame is a qualitative simulation. The Capture the Flag planner does the same: it simulates potential plans at some abstract level, then applies a static evaluation function to select the best plan. The static evaluation function incorporates such factors as relative strength and number of captured and threatened flags of both teams, to describe how desirable this future world state is.

Simulation is a costly operation, and in order to do it efficiently, Capture the Flag must be able to jump ahead to times when interesting events take place in the world. This is difficult because Capture the Flag takes place in continuous space and essentially continuous time. Naïve forward search is intractable because the search space is essentially infinite. Naïve decomposition of the state space into states, for instance, by laying a grid over the physical space or advancing time by pre-established large units, introduces a variety of pathologies. Our solution is to dynamically find state boundaries called *critical points*. Instead of advancing the world tick by tick, which is time-consuming, we jump right to the next critical point.

A critical point is a time during the execution of an action where a decision might be made, or the time at which it might change its behavior. If this decision can be made at any time during an interval, it is the *latest* such time.

Critical point search is illustrated in Figure 1, in which a white blob is considering attacking a black flag. The search is complicated by a black blob in the vicinity. It might defend its flag or attack the white flag. The white planner wishes to assess the outcome of its plan in both conditions. It could do this by advancing time forward in very small increments, expanding the entire state space of the interaction between the forces, but fortunately, it doesn't have to. The points marked CP in Figure 4 are critical points for white's plan. The first (the one closest to the white blob's current location) occurs at the last instant at which the black blob could move back to its flag and defend it successfully. The second occurs at the last instant at which the white blob could abandon its attack an scurry back to defend its flag if black attacks it. It is not necessary, to evaluate white's plan, to simulate the state of the game *except* at these points and the points at which black and white reach their own or opponent flags. Critical point search is very efficient and allows the planner to evaluate plans by straightforward minimax search.

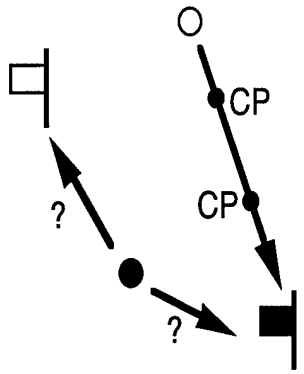


Figure 1. An illustration of critical points for search.

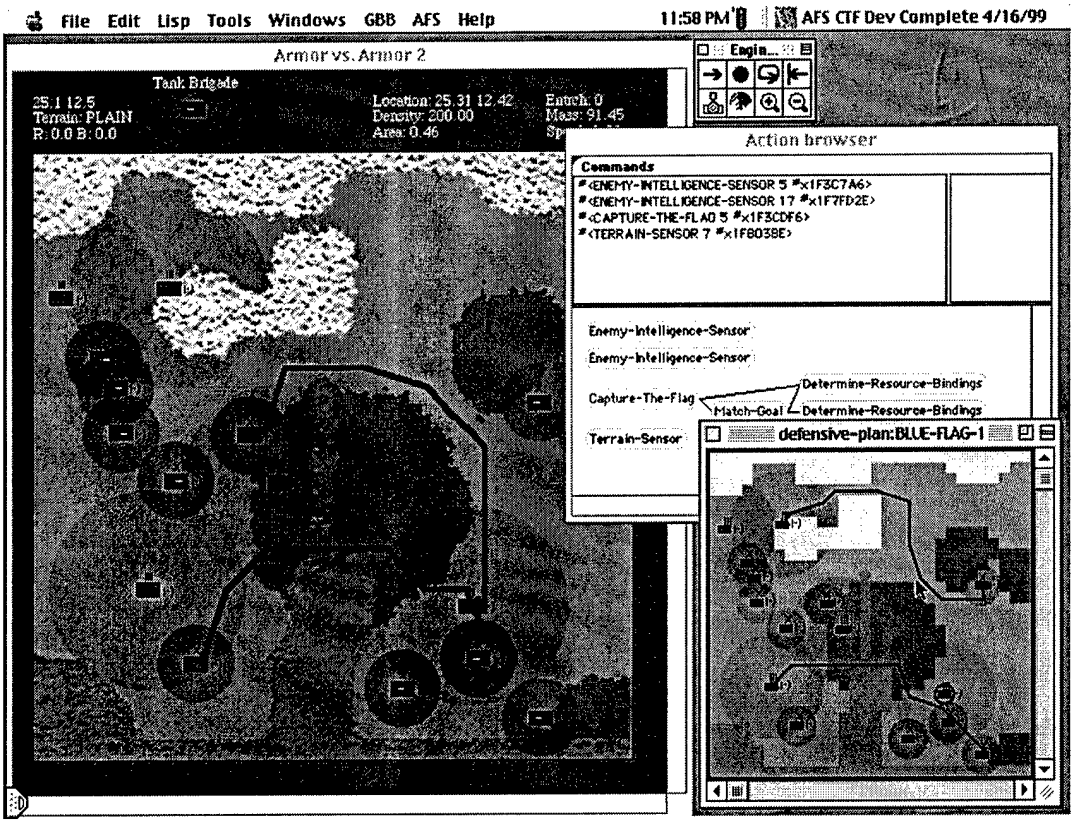


Figure 2. A screen capture of the Capture the Flag system

A screen image of the Capture the Flag system is shown in Figure 2. The leftmost panel shows the state of the game with forces arrayed on terrain. One can make out military symbols for these forces. The circles around each show the radius of direct fire – the region within which one blob can apply force to another. The arrows in the leftmost panel represent part of the Red planner's plan. A similar image in the lower right of the screen represents the Red planner's assessment of Blue's best actions. The other panel shows part of the plan hierarchy for the Red planner.

Maps for Verbs

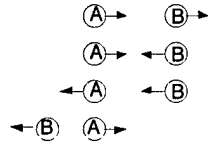
Much of what we know and say refers to the dynamics of our world. Here I include our mental world, the world of social interactions, and other not-entirely-physical environments. We have a large class of linguistic objects – verbs – devoted entirely to expressing dynamics. Subtle differences in the meanings of verbs, which linguists call "manner," are also often dynamical. For instance, the difference between "nudge" and

"shove" is partly a matter of mass, movement, and energy transfer from one body to another; and partly a matter of intention. Some AI researchers – those concerned with stochastic control, Markov decision processes, qualitative physics and the like – have developed representations of dynamics that machines can reason with. However, the knowledge representation community and ontology engineers seem satisfied with declarative statements *about* dynamics rather than representations *of* dynamics. They say, "Two agents collided and one fell down," but they don't describe the collision or the dynamics of falling. Ontologies generally describe everything about movement but the movement itself. Like a dictionary, they tell us that strolling is a casual, unhurried kind of walking, but they don't represent the actual movement.

Why should ontologies represent dynamics? Dynamical representations are compact in the sense that a single representation can describe dozens of related concepts. They make explicit the manner of movement and thus make fine distinctions between word meanings. They are grounded in the sense that one can attach sensors to a corpus of dynamical concepts and have the corpus recognize concepts from sensed movement – something no ontology can currently do (Rosenstein, Cohen, Schmill and Atkin, 1997). Dynamical representations of physical interactions are easily learned from observations of dynamics (Rosenstein et al., 1997) this is true also of dynamical representations of linguistic constructs (e.g., Regier, 1995; Elman, 1995). The strongest reason to consider dynamics as a foundation for ontologies, I think, is that the knowledge of the youngest humans – neonates and infants – is produced by interacting physically with the world.

Here we develop a dynamical representation of the meaning of verbs. The *distance* between A and B, D(A,B) is a projection of the not-necessarily physical locations of A and B onto a one-dimensional *progress space*. P(A) and P(B) are the locations of A and B in progress space and D(AB) = P(B) - P(A). Note that the transformation of the states of A and B to P(A) and P(B) may be quite complex, and it might not even be physical. For instance, when a chef says he's "halfway done" with a meal, he is transforming the remaining tasks to a representation of the time required to finish the meal; this requires knowledge and skill. And when a professor asserts that a student is "advanced" relative to others she is mapping some attributes of the students to an entirely metaphorical line. For every domain, we must be able to map the "locations" of A and B (whether spatial coordinates or locations in a metaphorical space) into P(A) and P(B).

Velocities for A and B are defined in terms of P(A) and P(B), in the usual way, namely, V(A) = dP(A)/dt. Acceleration is just the derivative of velocity, V'(A) = dV(A)/dt. In physical space, relative velocity depends not only on V(A) and V(B), but also on the angle of A's trajectory relative to B's. In progress space, however, A and B are always traveling along a line. Since A and B are arbitrarily assigned labels, there are just four qualitative kinds of interactions between A and B in progress space:



In the first, A is behind B, and both are moving in the same direction; the point of contact is no closer than the rightmost agent and D(AB) > 0. In the second, A and B are moving toward each other in progress space and the point of contact is between them; again, D(AB) > 0. The third situation has A and B moving in the same direction, but their velocities are negative relative to the first situation, D(AB) > 0, and the point of contact is not closer than the leftmost agent. In the fourth situation, no contact can occur; I will not discuss this case any further.

In the first qualitative interaction, above, we define $V(A) \geq 0$ and $V(B) \geq 0$; in the second, $V(A) \geq 0$ and $V(B) \leq 0$. In the third, $V(A) \leq 0$ and $V(B) \leq 0$. We define relative velocity,

$$VR = V(A) - V(B).$$

For instance, if A's velocity is 10cm/sec. and B's is 20 cm/sec., but B and A are moving toward each other along a line (i.e., the second qualitative interaction, above), then VR = V(A) –V(B) = 10 – (–20) = 30cm/sec. In the third qualitative interaction, above, VR = –30cm/sec.

The interaction of A and B can be plotted in a two-dimensional space, called a *map*, as shown in Figure 3. (Maps are also called phase portraits or phase diagrams; when the axes of a map represent values of a single variable measured at different times, maps are called delayed coordinate embeddings. Previous work in AI and Cognitive Science that uses maps as representations includes Rosenstein, et al, 1997; Bradley and Easley, 1997; Campbell and Bobick, 1995; Thelen and Smith, 1994) The horizontal dimension is D(AB), the distance from A to B. The vertical dimension is VR, the relative velocities of A and B. The horizontal midline represents equal velocity, V(A)=V(B). Above this midline, A is moving faster than B (or B is heading toward A, or both); below it, A is moving more slowly than B.

Some trajectories in this map are impossible. From the point labeled **a**, all trajectories must stay to the left of the vertical dashed line. This is because any vector from **a** to a point to the right of the line would mean A is slower than B but D(AB) = P(B) – P(A) is decreasing. This can happen only if P(A) is increasing faster than P(B), which is inconsistent with V(A) < V(B). The shaded semicircle represents forbidden trajectories. Similarly, at point **b**, no vector can point left of the dotted line, because such a vector would represent B gaining on A (or A falling back toward B) which is inconsistent with V(A) > V(B). At point **c**, the forbidden vectors flip from the left of the vertical line to the right, when A's velocity flips from being higher than B's to being lower.

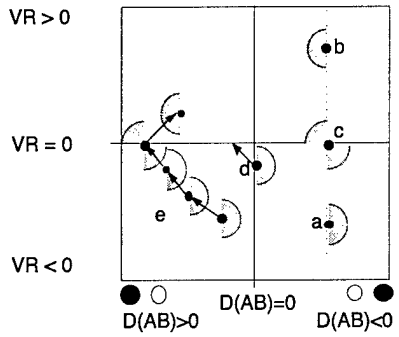
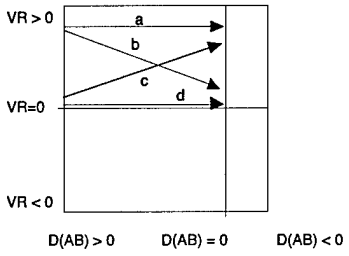


Figure 3 Only some trajectories are physically possible

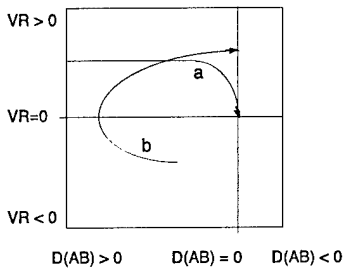
Point d illustrates that D(AB) and velocities may change simultaneously. Imagine the vector to represent one time step of arbitrary duration. At the beginning of this interval, P(A) = P(B) and B is moving faster than A. At the end of the interval, the velocities are equal but B is ahead of A.

The trajectory **e** shows five time steps of a "chase" behavior. In the first four steps, B is pulling away from A but at a decreasing rate, which is to say although A remains behind B, it speeds up relative to B, until, at the end of the fourth time step, the velocities are equal. At the end of the fifth time step, A's velocity exceeds B's, and A now starts to gain on B. One can imagine trajectory **e** continuing in a closed loop to the left of the line D(AB)=0, representing A repeatedly gaining on B then falling back, never catching B. A closed loop that crosses D(AB)=0 represents A and B "taking turns leading," like cyclists in a race.

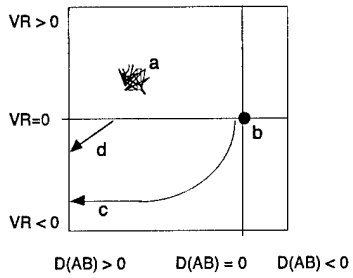
This framework has sufficient representational power to describe many interactions between A and B, as shown in the following examples.



a. V(R) stays constant and relatively high until contact. "A runs into B full-tilt"
b. VR decreases until contact: "touch,catch-up,"
c. Looks like a "hit," as A speeds up as it approaches B
d. "Drifting," barely moving toward each other because the relative velocity is nearly equal



a. Rapid deceleration, "hit the brakes."
b. Initially A is losing ground to B, then "makes up for lost time," "comes storming back," "recoups its losses," "B eludes A briefly," etc.



a. "B follows A, A leads B." Convoy, keeping close, etc.
b. A and B are touching, either at rest or at matched velocities. Contact.
c. "B narrowly escapes A" (because it started to move away from A very near the contact point)
d. "B avoids A" (because a small effort, well before imminent contact, puts B out of reach for A).

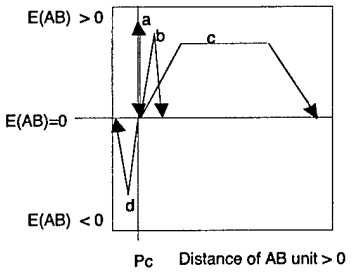
Admittedly, some aspects of interactions between A and B are not represented. The directions of physical movement of A and B are not captured, only their relative positions in progress space (i.e., P(A) and P(B)). Similarly, relative, not absolute velocities are represented. This means that the framework does not distinguish Case 1: A and B are moving in the same direction and A is catching B because of superior velocity; from Case 2: A and B are moving toward each other. Hence, we cannot differentiate "A catches B" from "A and B embrace." Nor can we distinguish subtle intentional relationships between A and B. Suppose A and B are moving in the same direction, with B in the lead, and with D(AB) varying in a narrow range. Is A trying to catch B while B tries to evade capture, or is A trying to follow B at a roughly constant distance?

An easily remedied representational deficit is that many verbs describe what happens when A and B make contact, whereas the previous examples all describe the interaction leading up to contact. Let us extend the framework to include types of contact.

THREE PHASES OF INTERACTIONS

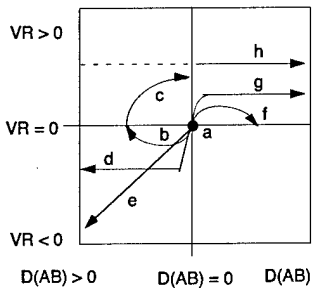
Physical interactions between agents can be viewed as having three phases. Consider the verb "push," for example. To push something, I first approach it and make contact with it. Generally, I try to achieve VR = 0 at D(AB) = 0, so that I gently touch the thing I'm trying to push. I apply force to it while remaining in contact for a period of time. When I or the thing I'm pushing breaks off contact, I may continue to move, or it may, or both. The three phases of a push, then, are **before**, **during**, and **after** contact. Many verbs of physical interaction can be represented in these terms; for example, a hit is like a push except that my velocity is high when I make contact and I stay in contact for a relatively short time. We have all received pushes that seemed a bit too much like hits; we might call them shoves. Where is the boundary between a push, a shove and a hit? There are no clear categorical boundaries: One's interpretation of an interaction depends on its dynamics, certainly, but also on contextual factors such as the intentions of the agents as described below.

Once contact has been made, and a pair of agents is in the **during** phase, the salient dynamics concern position and energy exchange. Note that we don't care about relative position (i.e., distance between A and B) because by definition D(AB)=0 in the **during** phase. Similarly, relative velocity must be zero, otherwise relative position would change. A dynamic map for the during phase has the distance of the AB unit from the point of contact Pc on the horizontal axis, and the transfer of energy from A to B on the vertical dimension. We view the interaction from the perspective of agent A and say E(AB)>0 if the net transfer of energy is to B, and E(A,B)<0 if B pushes harder.



a. A transfers a lot of energy to B without any movement: A crashes into a brick wall (B).
b. A transfers a lot of energy to B and the AB unit moves a little in the direction of A's movement. Pushing a car, a piano, or something else very massive.
c. A initially transfers no energy to B, but ramps up to a constant flow, then ramps down. A pushes B.
d. Like b except the AB unit moves in the direction of B's movement.

The denouement of the interaction between A and B is the **after** phase, which is entered when A and B break off contact. What seems most germane about this phase is the trajectories that A and B follow, so we could go back to the dimensions of **before** maps. A good reason to do so is that the **after** phase of one interaction may be the **before** phase of the next, especially for repetitive interactions such as tapping, hammering, harassing, and so on:



a. A and B remain at zero relative velocity and zero distance, attached.
b. B's velocity with respect to A increases, as does its distance from A, then relative velocity goes to zero, and A and B remain at a constant distance. As if A kicked, shoved, shunted or otherwise provided impetus for B.
c. Like b, except that A's velocity eventually increases again relative to B's, and the distance is reduced. This pattern would be observed in A hammering or harassing B.
d. A imparts some impetus to B and B maintains it. "Kickstart, jumpstart, get B going, initiate B's action, etc."
e. Like d except B keeps accelerating.
f. Curiously, contact with B increases, rather than decreases A's velocity and thus its position relative to B. "slingshot, boost, accelerate," etc.
g. Like f except achieving a constant relative velocity.
h. A's velocity relative to B is apparently unaffected by contact. One imagines the **before** trajectory as the dotted line. This is what we'd expect to see if A overtakes B without making contact, or if B is insubstantial (e.g., fog) and offers no resistance to A.

Now let's look at some combinations of **before**, **during**, and **after** phases. Illustrative trajectories from each phase are shown in the three panels of Figure 4. Each trajectory in each panel has a label, and complete trajectories through the triptych are denoted by three-letter sequences. For instance, **cah** denotes A approaching B at a constant, high speed; contact for zero time with zero energy transferred (the black dot at the origin of the **during** phase); then A moving away from B at the same high speed. This trajectory represents "A overtakes B."

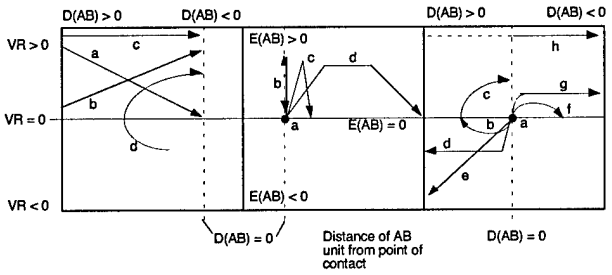


Figure 4. The **before, during** and **after** phases of physical interactions between A and B. The dashed vertical lines represent the point of contact, D(AB)=0. In the **before** and **after** phases, regions to the left of D(AB) = 0 represent A behind B and regions to the right represent A ahead of B. In the during phase, regions to the right of D(AB)

= 0 represent displacement of the AB unit (remaining in contact) from the point of contact.

Many verbs can be represented in this framework:

aaa A approaches B, touches it, and remains in contact with it. *A gently touches B* with no net transfer of energy between them. Relative velocity is inherently ambiguous: We know A and B have equal velocities in the **after** phase, but we don't know whether this velocity is zero.

ada A approaches B, makes contact, then gradually increases the energy it transfers to B, maintains a level of energy transfer, then ramps down. A and B remain in contact in the **after** phase. *A pushes B.*

adb **acb** A approaches B, makes contact, and gradually (d) or rapidly (c) increases the energy it transfers to B. In the **after** phase, B moves a little ahead of A. Initially its velocity increases relative to A's then decreases. Depending on the rate of energy transfer, the amount transferred, and the distance B moves in the **after** phase, this is *kick, nudge, shove, propel*, and so on.

The movement in the **before** phase is inherently ambiguous: We don't know whether A is moving toward B, B is moving toward A, or both. Similarly, the increasing distance between A and B in the **after** phase might occur because A stops moving (or slows down) but B continues, or because B stops and A is recoiled, or a combination of effects. Thus, **acb** represents *A bounces off B* as well as *kick, shove*, and so on. Similarly, **acb** represents *symmetric repulsion*, where A and B approach each other, make contact, then bounce away from each other.

aca **ada** As above, except B doesn't move. Depending on rates and amounts of energy transferred, this too may be a *kick* or a *bump* (but not a *shove* or *propel*, because B doesn't move). Alternatively, **ada** denotes a more gradual interaction, as in *A leans against B, A strains against B.*

bce Whereas **b** in the **after** phase represents A and B moving apart with an increasing, then decreasing, velocity, trajectory **e** represents A and B moving apart with a strictly *increasing* velocity. Imagine a hand (A) pushing a cup (B), off the edge of a table. Or we might say A *dislodges* B, or *frees* it from some stricture. Or B might *flee* from contact with A.

cba A and B converge at a high, constant rate. At the instant of contact they exchange a lot of energy, and remain in contact during the **after** phase. This is what happens when a car *crashes* into a tree. More benignly, B may absorb all A's energy with no ill effect, but I know no verb to describe this interaction.

dcc This is a cyclic interaction where A and B converge, energy is transferred, and during the after **phase**, A and B diverge then converge again. Many verbs denote this repetitive pattern: *Hammer, harrass, clap*, and so on.

bbf A accelerates relative to B until the point of contact, B absorbs energy from A, and A is slowed down and eventually comes to rest a little beyond

B. *A pushes through B.*

bbg Like **bbf**, except A maintains a constant velocity after interacting with B. *A breaks free of B.*

As with the individual maps, this triptych represents many aspects of interactions but fails to represent others. Some ambiguities have already been discussed (see **adb** and **acb**, above). Because this framework doesn't represent actual spatial coordinates, it cannot differentiate the cases in Figure 5. Similarly, we cannot tell whether A is pushing an unyielding B, or A and B are pushing against each other. Another source of ambiguity arises because the framework doesn't specify what kinds of things A and B are. In particular, it is unclear what kind of energy A transfers to B and where this energy comes from. If A transfers kinetic energy, then the sequence **ab...** will in some cases be physically impossible because once the relative velocity of A and B reaches zero, there is no kinetic energy to transfer. On the other hand, if A is capable of generating movement itself, as most agents are, then it can transfer kinetic energy to B even after their velocities are matched, simply by increasing its velocity. Another ambiguity arises because no scales are specified in the maps. We can say one interaction involves more force than another (e.g., a shove versus a tap), but if we have only one trajectory and cannot calibrate it against others, then we cannot judge whether it is gentle or violent.

Despite these and other limitations in representational power, the framework is compact and simple, yet captures many verb meanings. Finer distinctions in meaning can be had by adding dimensions to the maps (e.g., x and y spatial dimensions). There is obviously a tradeoff between the expressivity and complexity of the maps.

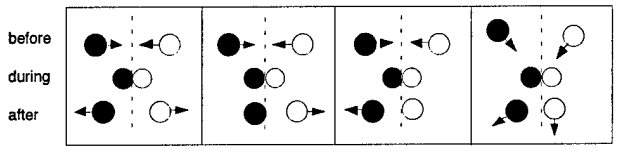


Figure 5. The framework cannot differentiate cases in which A and B recoil mutually; B is propelled but A doesn't move; A bounces off B; or A and B bounce off each other at unspecified angles.

A Study of Knowledge Reuse

We report an empirical study of knowledge reuse[1] By comparing the efforts of two HPKB groups under different conditions, we find that prior knowledge in the form of ontologies does help, though many factors affect how much it helps. This work also introduces metrics and methods for evaluating the contribution of prior knowledge to knowledge-based systems.

---

[1] Does Prior Knowledge Facilitate the Development of Knowledge-based Systems? Paul Cohen, Vinay Chaudhri, Adam Pease, Robert Schrag, AAAI 1999.

By *prior knowledge* we mean the knowledge one has available in an ontology or knowledge base prior to developing a knowledge-based system. Several large ontologies have been developed including Cyc (Lenat,1995), Sensus (Knight, 1994), Ontolingua (Farquhar, 1996). All these systems contain hierarchies of knowledge. At the upper levels, one finds knowledge that is general to many applications, such as knowledge about movement, animate agents, space, causality, mental states, and so on. The lower levels contain knowledge specific to domains; for example, rules for inferring the effects of tactical military operations. Bridging general and specific knowledge, one finds middle-level knowledge (Lenat and Guha, 1990); collections of terms and axioms about phenomena such as human physiology, more general than a particular medical expert system but less general than, say, knowledge about physical systems. In addition to hierarchies of terms, all the ontologies cited above contain *axioms* or *rules*, for instance, "if x is an educational institution then x pays no taxes"; and inference methods such as resolution or more specialized forms of theorem-proving. Axioms and rules confer a functional kind of *meaning* on the terms they contain, that is, the meaning of a term is the things one can legitimately say (infer) about it.

One claim of ontologists is that it is easier to build a domain-specific knowledge base **KB** inside an ontology **O**, or informed by **O**, than without **O**. Some of the ways that **O** can help are illustrated in Figure 6. First, a term **p** that you wish to add to **KB** might already exist in **O**, saving you the trouble of adding it. Second, axioms or rules relating to **p** might already exist in **O**, saving you the trouble of thinking of them and encoding them. Third, within **O**, **p** might be a subclass of **v**, so you also have the benefit of axioms about **v** inherited through **p**.

Now suppose you want to add a concept **p'** to **KB**, and **p'** is not exactly **p**, but is similar in some respects. For instance, **p** might be part of a microtheory about economics, and **p'** might belong to a microtheory about fluid flows, but both **p** and **p'** represent the concept "source." More generally, suppose the *structure* of the theory of economics in **O** parallels the structure of the theory of fluids that you are trying to build in **KB**. Thus, a fourth way that **O** can help you to build **KB** is to help you structure the theory in **KB**. Designing the structure of microtheories is very time consuming, so this kind of help may be the most important of all.

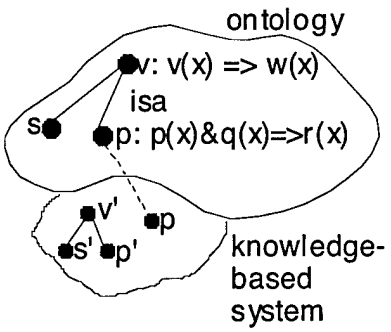


Figure 6. Some ways an ontology O can help one build a knowledge base KB.

Unfortunately it is difficult to assess experimentally how the structure of **O** helps one build **KB**s with similar structure, so we focus here on the first three ways that **O** can help one build **KB**.

A Metric

Suppose one wishes to add an axiom, "If x is a nation then x maintains an army," to **KB**. This axiom contains three terms, **nation**, **maintains**, and **army**. Suppose the first two terms already exist in **O** but **army** does not. As two thirds of the terms required to add the axiom to **KB** exist in **O,** we say the *support* provided by **O** in this case is 2/3. In general, every item i one wishes to add to **KB** contains n(i) terms, k(i) of which are already in **O,** and support is s(i)=k(i)/n(i). Of course, adding **army** to **O** changes **O**, and the support offered by **O** for future axioms might be higher because **army** was added. Thus, support is indexed by versions of the ontology: s(i,j)=k(i,j)/n(i) is the support provided by version **O**j of the ontology for concept i.

In the following section we analyze how prior ontology – what was available before SQs, TQA and TQC were released – supported the development of the Teknowledge and SAIC systems. The former system was based on Cyc, and much of its development was done at Cycorp, so we call it Cyc/Tek here. The SAIC system was a collection of component systems, none of which answered all the questions in any test batch. The one we analyze here, developed by SRI International, answered roughly 40 of the 110 questions in each batch; we lack data for the other components of the SAIC system. To compare the Cyc/Tek and SRI systems properly we will report two sets of results for Cyc/Tek, one for all the test questions and another for the subset of questions answered by the SRI system.

The Cyc/Tek and SRI systems also differed in the prior ontologies available to them. Long before testing began, Cycorp, the developers of Cyc, released their *upper ontology* (UO), which contains very general class names; subclass relationships; instance-type relationships; relation names and their argument types; function names, their argument types, and the types of value they return; as well as English documentation of every class, function and relation; and a mapping to terms in the Sensus ontology developed by ISI. Whereas the SRI team had access to the UO, only, Cyc/Tek had access to all of Cyc.

Results

The performance of the Teknowledge and SAIC integrated systems is analyzed in (Cohen et al., 1998). Performance is not the focus of this paper – support provided by ontologies is – but two performance results set some context for the following discussion of support and reuse: Both systems performed better on the sample questions (SQs) than on TQA, and both performed better when re-tested on TQA and TQC than on the corresponding tests performed four days earlier. In the four days between test and retest, significant improvements were made to the systems. The question is, how much did the prior ontologies help in making these improvements?

We present results for two kinds of knowledge development. One is the development of knowledge sufficient to encode in a formal language the test questions in each batch, the other is the development of knowledge to answer the test questions. Results for the former are summarized in Table 1. The columns of the table represent the SRI system, which was tested on roughly 40 questions in each batch of 110; the Cyc/Tek system tested on the same questions as the SRI system; and the Cyc/Tek system tested on all 110 questions in each batch. Three numbers are reported for each system: n is the number of terms needed to encode all the questions attempted (i.e., roughly 40 or 110); k is the number of terms available in a prior ontology; and s is the ratio of k to n. The rows of Table 1 represent the batches of questions and the help provided by different prior ontologies.

For example, SQ | UO means "the help provided by the upper ontology (UO) in encoding the sample questions (SQ)." One can see in this row that SRI needed 104 terms to encode roughly 40 of the sample questions, and 22 of these terms were found in the UO, so the help provided by the UO is 22/104 =.21. Encoding the questions in SQ required a number of terms to be added to the ontologies, and these terms were available to help encode questions in TQA and TQC. The notation TQA | UO denotes the help provided by the UO *only*, whereas TQA | SQ denotes the help provided by *everything encoded up through* SQ. Similarly, TQC | TQA denotes the help provided in encoding the questions in TQC by the terms in the ontology including those defined for SQ and TQA. For the Cyc/Tek system, our data support only a simpler distinction, between UO terms and non-UO terms, the latter category including the entire Cyc ontology and all terms defined while encoding the test questions. The category of non-UO terms is reported in rows labeled "Cyc" in Table 1. For instance, 292 terms were required by Cyc/Tek to encode the 110 questions in TQA, 289 of them were available in Cyc, including some defined when the sample questions SQ were added. Note that SRI used only the public release of the upper ontology, so all rows in which questions were encoded with the help of Cyc are marked n/a for SRI.

	SRI			Cyc/Tek(40)			Cyc/Tek(110)		
	n	k	s	n	k	s	n	k	s
SQ | UO	104	22	.21	143	60	.42	246	97	..39
SQ | Cyc	n/a	n/a	n/a	143	105	.73	246	182	.73
TQA | UO	104	20	.19	150	67	.45	292	118	.40
TQA | SQ	104	81	.78	n/a	n/a	n/a	n/a	n/a	n/a
TQA | Cyc	n/a	n/a	n/a	150	150	1.0	292	289	.98
TQC | UO	106	16	.15	153	71	.46	304	117	.38
TQC | TQA	106	82	.77	n/a	n/a	n/a	n/a	n/a	n/a
TQC | Cyc	n/a	n/a	n/a	153	153	1.0	304	395	.98

Table 1. Support (s) provided by ontologies for the task of encoding test questions.

The six reuse rates from Table 1 are presented graphically in Figure 7. Reuse from the UO on all test question batches clusters in the lower half of the graph. The highest levels of reuse from the UO are achieved by Cyc/Tek on the roughly 40 test questions encoded by SRI. The upper half of the graph represents reuse from the UO *and* all of Cyc in the Cyc/Tek conditions; and reuse of terms defined for earlier test question batches, in the SRI condition.

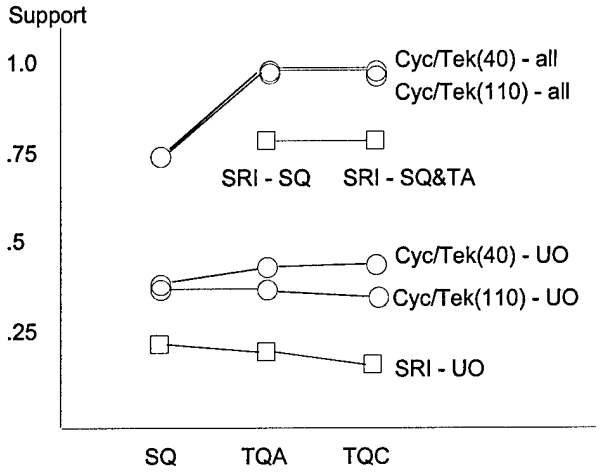


Figure 7. Support rates for SRI and Cyc/Tek. Lines denoted "UO" represent reuse of terms from the upper ontology. SRI-SQ denotes SRI's reuse of terms from the UO and the SQ-encoding effort; SRI-SQ&TA adds in terms defined during the TA-encoding effort. Cyc/Tek(40)-all and Cyc/Tek(110)-all denote reuse of terms from all of Cyc.

Cyc/Tek had higher support numbers in all conditions than SRI, meaning they reused more terms in their prior ontologies than SRI did. However, we have broken the data into support provided to Cyc/Tek by *all* of Cyc vs. support provided by just the upper ontology, which SRI had. For example, the first row of Table 1 shows that to encode roughly 40 sample questions, SRI required 104 terms of which it found 22 in the UO; whereas Cyc/Tek required 143 terms to encode the *same* questions, and found 60 in the UO. Similarly, Cyc/Tek required 246 terms to encode all 110 sample questions, and found 97 in the UO.

Cyc/Tek required slightly more terms to encode test questions (2.86 terms/question) than SRI (2.62 terms/question), and got more support from prior ontologies. For example, for Cyc/Tek to encode the roughly 40 questions in the TQA batch that SRI encoded, they required 150 terms, all of which existed in the Cyc ontology.

In one respect, the SRI and Cyc/Tek results are very similar. The reuse rate of terms *not* in the upper ontology – terms in Cyc or terms developed for earlier batches of test questions – was 55%-60% for both SRI and Cyc/Tek, across question batches TQA and TQC. This result is shown in Table 2. The columns in this table represent the number of terms needed to encode a test batch, N; the number found in the upper ontology, K(UO);

the number found elsewhere, K(other); and the ratios of K(UO) and K(other) to N. That is, the support provided by terms in the upper ontology is s(UO)=K(UO)/N, while the support provided by other prior ontology is s(other)=K(other)/N. Note that s(other) ranges from .54 to .62 for test batches TQA and TQC. (Cyc/Tek also found support for coding up the SQ questions from parts of Cyc other than UO; these support figures are .31 and .40 for the 40 and 110 test questions, respectively.) For TQA and TQC, the overall rates of reuse of non-UO terms for Cyc/Tek and SRI were .58 and .60, respectively; whereas the overall reuse of UO terms for Cyc/Tek and SRI was .41 and .17, respectively. Thus, much of the difference in reuse statistics between SRI and Cyc/Tek is due to their exploitation of the upper ontology. Said differently, 22% of the terms SRI reused came from the upper ontology while the figure was 42% for Cyc/Tek.

	N	K(UO)	K(other)	S(UO)	S(other)
SRI TQA	104	20	61	.19	.59
SRI TQC	106	16	66	.15	.62
Cyc/Tek SQ(40)	143	60	45	.42	.31
Cyc/Tek TQA(40)	150	67	83	.45	.55
Cyc/Tek TQC(40)	153	71	82	..46	.54
Cyc/Tek SQ(110)	246	97	85	.39	.40
Cyc/Tek TQA(110)	292	118	171	.40	..58
Cyc/Tek TQA(110)	304	117	185	.38	.60

Table 2. Support provided by terms in UO and terms from other prior knowledge bases and ontologies for the task of encoding test questions.

In addition to encoding test questions, Cyc/Tek and SRI developed knowledge to answer the questions. This knowledge, called *axioms* generically, is composed of terms, so we can ask how prior ontologies helped the development of axioms. As before the relevant metric is s(i,j)=k(i,j)/n(i), only here, n(i) denotes the number of terms required to encode the ith axiom.

SRI provided data on how ontologies supported writing axioms. The rows of Table 3 represent the phases of the experiment and the source of prior ontology. The first row, SQ | UO shows that 1703 axioms were encoded to solve the sample questions SQ, and these axioms required 461 terms, of which 51 were in the upper ontology, UO, for a support value of 0.11. The second row shows that in the four days between the test and retest on batch TQA, 123 axioms were encoded, requiring 195 terms. 30 of these terms were found in the UO. The third row shows that 109 of the 195 terms were found in *all* the ontology developed prior to the test on TQA, namely UO *and* SQ. A comparison of the second and third rows shows that 109–30=79 reused terms came from SQ. The same pattern repeats in the two remaining phases of the experiment: After the scenario modification but before TQC, 1485 axioms were added to the SRI system. These

required 583 terms of which 40 existed in the UO and 254 were found in the UO, SQ, and TQA prior ontologies. Similarly, between the test and retest on TQC, 215 terms were required for 304 axioms; only 24 of these existed in the UO, and 100 more were found in the ontologies developed after the UO.

It is unclear why prior ontologies provided significantly less support for encoding axioms than for encoding test questions. In both cases the support came in the form of terms, but why are the terms required to define axioms less likely to be in a prior ontology than the terms needed for test questions? One possibility is that test questions include fewer terms that represent *individuals* (e.g., #$HassiMessaoud-Refinery) than do axioms, so terms in test questions are less specific and more likely to exist in a prior ontology than terms in axioms. We will be looking at our data more closely to see whether this is the case.

	SRI			
	Axioms	n	k	s
SQ | UO	1703	461	51	.11
From TQA to TQA retest | UO	123	195	30	.15
From TQA to TQA retest | SQ	123	195	109	.56
From TQA retest to TQC | UO	1485	583	40	.09
From TQA retest to TQC | TQA	1485	583	254	.44
From TQC to TQC retest | UO	304	215	24	.11
From TQC to TQC retest | TQC	304	215	124	.58

Table 3: SRI measured the number of terms required to add problem-solving axioms to their system, and the reuse of terms from the UO and subsequent ontology efforts.

Discussion

Does prior knowledge in ontologies and domain-specific knowledge bases facilitate the development of knowledge-based systems? Our results suggest that the answer depends on the kind of prior knowledge, who is using it, and what it is used for. The HPKB upper ontology, 3000 very general concepts, was less useful than other ontologies, including Cyc and ontologies developed specifically for the crisis management domain. This said, Cyc/Tek made more effective use of the upper ontology: 42% of the terms it reused came from there whereas 22% of the terms SRI reused came from the upper ontology. Why is this? One reason is probably that Cycorp developed the upper ontology and was more familiar with it than SRI. Knowledge engineers tend to define terms for themselves if they cannot quickly find the terms in an available ontology. Once this happens – once a term is defined anew instead of reused – the knowledge base starts to diverge from the available ontology, because the new definition will rarely be identical with the prior one. Another reason for disparity in reuse of the upper ontology is that SRI preferred their own definitions of concepts to the available ones.

As to the uses of prior knowledge, our data hint at the possibility that prior knowledge is less useful for encoding axioms than it is for encoding test questions.

Whereas reuse of the upper ontology depends on who is using it, other ontologies seem to account for a roughly constant (60%) rate of reuse, irrespective of who developed these ontologies. For SRI, these ontologies were just those developed for batches of questions SQ, TQA, TQB, TQC and TQD. To be concrete, 62% of the terms required for TQC were defined while encoding SQ, TQA and TQB. The picture is a bit cloudier for Cyc/Tek because they had the Cyc ontology throughout, and we have not yet analyzed whether the overall 60% non-UO reuse came from terms defined for previous batches or from Cyc.

Despite this ambiguity we speculate that in the process of building a domain-specific knowledge-based system, the rate of reuse of terms defined earlier in the process is roughly 60%. Although the rate of reuse of terms from very general ontologies may be significantly lower (e.g., 20%–40%), the real advantage of these ontologies probably comes from helping knowledge engineers organize their knowledge bases along sound ontological lines. It is essential for the ontology community to collect data on this use of general ontologies.

References

Bradley, L and Easley, M. 1997. *Reasoning about sensor data for automated system identification.* In Advances in Intelligent Data Analysis. X. Liu, P. R. Cohen and M. Berthold, Eds. Lecture Notes in Computer Science, 1280. Springer. pp. 561–572.

Campbell, L. and Bobick, A. 1995. *Recognition of human body motion using phase space constraints*. MIT Media Laboratory Perceptual Computing Section Technical Report No. 309.

Paul Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Gunning, and Murray Burke. The DARPA High Performance Knowledge Bases Project. AI Magazine, Winter, 1998. pp. 25-49

Farquhar, A. Fikes, R., Rice, J. 1997. A collaborative tool for ontology contruction. International Journal of Human Computer Studies. Vol. 46. pp 707 - 727

Knight, K., and Luk, S. 1994. Building a large-scale knowledge base for machine translation. AAAI-94.

Lenat D. 1995. Artificial Intelligence. Scientific American. September.

Lenat D. and Guha R. 1990. Building Large Knowledge based system. Reading MA Addison Wesley.

Patil, R., Fikes, R., Patel-Schneider, P., Mackay, D., Finin, T., Gruber, T., and Neches, R. 1992. The DARPA Knowledge Sharing Effort: Progress Report. In KRR-92. pp 777 – 788.

Elman, J. 1995. *Language as a dynamical system.* In *Mind As Motion*, R. F. Port and T. van Gelder, Eds. MIT Press. pp. 196 – 225.

Lakoff, G. 1984. *Women, Fire and Dangerous Things*. University of Chicago Press.

Lakoff, G. and Johnson, M. 1980. *Metaphors We Live By*. University of Chicago Press.

Regier, T. 1995. A model of the human capacity for categorizing spatial relationships. *Cognitive Linguistics*, 6 (1), pp. 63 – 88.

Michael Rosenstein, Paul R. Cohen, Matthew D. Schmill, Marc S. Atkin. 1997. *Action, representation, prediction and concepts*. Presented at the 1997 AAAI Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control. Also available from `http://eksl-www.cs.umass.edu:80/publications-2.html`

Thelen, E. and Smith, L.B. 1995. *A Dynamic Systems Approach to the Development of Cognition and Action.* MIT Press.

Publications Supported By This Contract

Oates, Tim Matthew Schmill, Paul Cohen, Casey Durfee Efficient Mining of Statistical Dependencies 1999 AI & Statistics 99 Workshop, pp. 278-282 & Working Notes, pp. 133-141 http://eksl.cs.umass.edu/papers/oates-aistats99.pdf

Rosenstein, Michael Paul Cohen Continuous Categories for a Mobile Robot 1999 IJCAI 99 Working Notes, pp. 47-53 http://eksl.cs.umass.edu/papers/rosenstein-aaai99.pdf

Cohen, Paul Mary Litch What are Contentful Mental States? Dretske's Theory of Mental Content Viewed in the Light of Robot Learning and Planning Algorithms 1999 AAAI 99 Proceedings, pp. 108-113 http://eksl.cs.umass.edu/papers/cohen-aaai99a.pdf

Sebastiani, Paola Marco Ramoni, Paul Cohen, John Warwick, James Davis Discovering Dynamics using Bayesian Clustering 1999 Lecture Notes in Computer Science, pp. 199-209 http://eksl.cs.umass.edu/papers/sebastianiIDA99.pdf

Firoiu, Laura Tim Oates, Paul Cohen Learning Regular Languages from Positive Evidence 1998 CSS 98 Proceedings, pp. 350-355 http://eksl.cs.umass.edu/papers/firoiuCOGSCI98.pdf

Cohen, Paul Dynamic Maps as Representations of Verbs 1998 ECAI 98 Proceedings, pp. 145-149 http://eksl.cs.umass.edu/papers/cohen-ecai98.pdf

Piater, Justus Paul Cohen, Xiaoqun Zhang, Michael Atighetchi A Randomized ANOVA Procedure for Comparing Performance Curves 1998 ICML 98 Proceedings, pp. 430-438 http://eksl.cs.umass.edu/papers/ML98.pdf and piaterML97.pdf

Schmill, Matthew Michael Rosenstein, Paul Cohen, Paul Utgoff Learning What is Relevant to the Effects of Actions for a Mobile Robot 1998 Agents 98 Proceedings, pp. 247-253 http://eksl.cs.umass.edu/papers/schmill-aa98.pdf

Firoiu, Laura Tim Oates, Paul Cohen Learning a Deterministic Finite Automaton with a Recurrent Neural Network 1998 International Colloquium on Grammatical Inference Proceedings, pp. 90-101 http://eksl.cs.umass.edu/papers/firoiu-FIC98ab.pdf

Cohen, Paul Tim Oates A Dynamical Basis for the Semantic Content of Verbs 1998 AAAI 98 pp. 5-8 http://eksl.cs.umass.edu/papers/oates-aaai98-word.pdf

Oates, Tim David Jensen, Paul Cohen Discovering Rules for Clustering and Predicting Asynchronous Events 1998 AAAI 98 pp. 73-79 http://eksl.cs.umass.edu/papers/oates-aaai98-time.pdf

McGeoch, C.C. Paul Cohen, Doina Precup How to Find Big-Oh in Your Data Set (and How Not To) 1997 AI & Statistics 97 Preliminary Papers, pp. 347-354 http://eksl.cs.umass.edu/papers/McGeochIDA97.pdf and cohen-ais96a.pdf

Schmill, Matthew Tim Oates, Paul Cohen Learned Models for Continuous Planning 1999 AI & Statistics 99 Proceedings, pp. 278-282 http://eksl.cs.umass.edu/papers/schmill-ais99.pdf

Cohen, Paul Vinay Chaudrhi, Adam Pease, Robert Schrag Does Prior Knowledge Facilitate the Development of Knowledge-Based Systems? 1999 AAAI 99 Proceedings, pp. 221-226 http://eksl.cs.umass.edu/papers/cohen-aaai99.pdf

Oates, Tim Laura Firoiu, Paul Cohen Clustering Time Series with Hidden Markov Models and Dynamic Time Warping 1999 IJCAI 99 Working Notes, pp. 17-21 http://eksl.cs.umass.edu/papers/oates-ijcai99SL.pdf

Sebastiani, Paola Marco Ramoni, Paul Cohen Unsupervised Classification of Sensory Input in a Mobile Robot 1999 IJCAI 99 Working Notes, pp. 23-28 http://eksl.cs.umass.edu/papers/sebastiani-ijcai99wksp.pdf

Firoiu, Laura Paul Cohen Abstracting from Robot Sensor Data using Hidden Markov Models 1999 ICML 99 Proceedings, pp. 23-28 or pp. 106-114 http://eksl.cs.umass.edu/papers/firoiu-icml99.pdf

Cohen, Paul Deep Understanding of Courses of Action 1999 DARPA Workshop on Knowledge Based Planning for Coalition Forces

Schmill, Matthew Tim Oates, Paul Cohen Learning Planning Operators in Real-World, Partially Observable Environments 2000 AIPS 00 Proceedings, pages 246-253. AAAI Press. http://eksl.cs.umass.edu/papers/schmill-aips00.pdf

Cohen, Paul Marco Ramoni, Paola Sebastiani Unsupervised Clustering of Robot Activities: A Bayesian Approach 2000 Agents 00 http://eksl.cs.umass.edu/papers/cohenIAS00a.pdf and cohen-agents00.pdf

Cohen, Paul Growing Ontologies 1999 ECCS 99 Proceedings, http://eksl.cs.umass.edu/papers/cohen-ijcai99.ps and cohen-aaai98.pdf and 98-20.pdf

Oates, Tim Identifying Distinctive Subsequences in Multivariate Time Series by Clustering 1999 KDD 99 Proceedings, pp 322-326

http://eksl.cs.umass.edu/papers/oates-kdd99.pdf and
oates-icml99.ps

Provost, Foster David Jensen, Tim Oates Efficient Progressive Sampling 1999 KDD 99 Proceedings, http://eksl.cs.umass.edu/papers/jensen-kdd99.pdf

Oates, Tim Identifying Qualitatively Different Outcomes of Actions: Experiments with a Mobile Robot 1999 IJCAI 99 Proceedings, pp. 30-36 http://eksl.cs.umass.edu/papers/oatesijcai99.pdf

Sebastiani, Paola Marco Ramoni, Paul Cohen Bayesian Clustering of Sensory Inputs by Dynamics 1999 http://eksl.cs.umass.edu/papers/sebastiani-ijcai99.ps

Rosenstein, Michael Paul Cohen Continuous Categories for a Mobile Robot 1999 AAAI 99 Proceedings, pp. 634-640 http://eksl.cs.umass.edu/papers/rosenstein-aaai99.pdf

Oates, Tim Zachary Eyler-Walker, Paul Cohen Using Syntax to Learn Semantics: An Experiment in Language Acquisition with a Mobile Robot 1999 http://eksl.cs.umass.edu/papers/oates-cogsci99.pdf

Schmill, Matthew A Distributed Approach to Finding Complex Dependencies in Data 1998 http://eksl.cs.umass.edu/papers/98-13.ps and http://eksl.cs.umass.edu/papers/schmill-cikm98-data.ps and schmill-kdd-data.ps

Ramoni, Marco Paola Sebastiani, Paul Cohen, John Warwick, James Davis Bayesian Clustering by Dynamics 1999 Machine Learning Journal L. Giles and R. Sun (Eds.), Sequence Learning: Paradigms, Algorithms, and Applications. Springer, New York, NY. http://kmi.open.ac.uk/kmi-abstracts/kmi-tr-89-abstract.html

Cohen, Paul Evaluation Working Group: Plan for May 1998 1997 HPKB

Jensen, David Yulin Dong, Barbara Staudt Lerner, Leon Osterweil, Stanley Sutton, Alexander Wise Representing and Reasoning about Knowledge Discovery Processes 1999 WACC http://eksl.cs.umass.edu/papers/jensen-wacc99.ps

King, Gary Warren Tim Oates The Importance of Being Discrete: Learning Actions Through Interaction 2000 http://eksl.cs.umass.edu/papers/00-15.pdf

Firoiu, Laura Rule Induction from Noisy Examples 2000 AAAI 00 http://eksl.cs.umass.edu/papers/firoiu-aaai00.pdf

Oates, Tim Matthew Schmill, Paul Cohen, Casey Durfee Efficient Mining of Statistical Dependencies 1999 IJCAI 99 Proceedings, pp. 794-799 http://eksl.cs.umass.edu/papers/oates-aistats99.pdf

McGeoch, C.C. Paul Cohen, Doina Precup How to Find Big-Oh in Your Data Set (and How Not To) 1997 IDA 97 Proceedings, pp. 41-52 McGeochIDA97.pdf and http://eksl.cs.umass.edu/papers/cohen-ais96a.pdf

Sebastiani, Paola Marco Ramoni, Paul Cohen, John Warwick, James Davis Discovering Dynamics using Bayesian Clustering 1999 IDA 99 Proceedings, pp. 199-209 http://eksl.cs.umass.edu/papers/sebastianiIDA99.pdf

King, Gary W. Tim Oates The Importance of Being Discrete: Learning Classes of Actions and Outcomes through Interaction 2001 AI 01

Ramoni, Marco Paola Sebastiani, Paul Cohen Bayesian Clustering by Dynamics 2001 Machine Learning Jan-01 http://eksl.cs.umass.edu/papers/ramonimac.pdf

King, Gary LIFT - The LIsp Framework for Testing 2001 http://eksl.cs.umass.edu/papers/01-25.pdf

Gary King Tim Oates The Importance of Being Discrete: Learning Classes of Actions and Outcomes through Interaction 2001 Proceeding of the Fourteenth Canadian conference on ARtificial Intelligence http://www-eksl.cs.umass.edu/papers/kingai01.pdf

Addendum. Report on All HPKB Performers Research.

The DARPA High Performance Knowledge Bases Project

Paul Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Easter, David Gunning, Murray Burke[2]

Abstract: Now completing its first year, the High Performance Knowledge Bases project promotes technology for developing very large, flexible and reusable knowledge bases. The project is supported by the Defense Advanced Research Projects Agency and includes more than fifteen contractors in universities, research laboratories and companies. The evaluation of the constituent technologies centers on two challenge problems, in crisis management and battlespace reasoning, each demanding powerful problem solving with very large knowledge bases. This article discusses the challenge problems, the constituent technologies, and their integration and evaluation.

Introduction

Although a computer has beaten the world chess champion, no computer has the common sense of a six year old child. Programs lack knowledge about the world sufficient to understand and adjust to new situations as people do. Consequently, programs have been poor at interpreting and reasoning about novel and changing events such as international crises and battlefield situations. These problems are more open-ended than chess. Their solution requires shallow knowledge about motives, goals, people, countries, adversarial situations, and so on; as well as deeper knowledge about specific political regimes, economies, geographies, and armies.

The High Performance Knowledge Base (HPKB) program is sponsored by the Defense Advanced Research Projects Agency (DARPA) to develop new technology for knowledge based systems. It is a three-year program, ending in FY 99, with funding totaling 34 million dollars. HPKB technology will enable developers to rapidly build very large knowledge bases – on the order of 10^6 rules, axioms, or frames – enabling a new level of intelligence for military systems. These knowledge bases should be comprehensive and reusable across many applications, and they should be easily maintained and modified. Clearly these goals require innovation in many areas, from knowledge representation to formal reasoning and special purpose problem solving, from

[2] The authors wish to thank Stuart Aitken, Vinay Chaudhri, Cleo Condoravdi, Jon Doyle, John Gennari, Yolanda Gil, Moises Goldszmidt, William Grosso, Mark Musen, Bill Swartout and Gheorghe Tecuci for their help preparing this article.

Paul Cohen

knowledge acquisition to information gathering on the Web to machine learning, from natural language understanding to semantic integration of disparate knowledge bases.

For roughly one year, HPKB researchers have been developing knowledge bases containing tens of thousands of axioms concerning crises and battlefield situations. Recently the technology was tested in a month-long evaluation involving sets of open-ended test items, most of which were similar to sample (training) items but otherwise novel. Changes to the crisis and battlefield scenarios were introduced during the evaluation to test the comprehensiveness and flexibility of knowledge in the HPKB systems. The requirement for comprehensive, flexible knowledge about general scenarios forces knowledge bases to be large. Challenge problems, which define the scenarios and thus drive knowledge base development, are a central innovation of HPKB. This article discusses HPKB challenge problems, technologies and integrated systems, and the evaluation of these systems.

The challenge problems require significant developments in three broad areas of knowledge-based technology. The overriding goal of HPKB — to be able to select, compose, extend, specialize, and modify components from a library of reusable ontologies, common domain theories, and generic problem-solving strategies — is not immediately achievable, and requires some research into foundations of very large knowledge bases, particularly research in knowledge representation and ontological engineering. Then there is the problem of building on these foundations to populate very large knowledge-bases. The goal is for collaborating teams of domain experts (who may lack training in computer science) to easily extend the foundation theories, define additional domain theories and problem solving strategies, and acquire domain facts. Knowledge is not enough, of course; one also requires efficient problem solving methods. HPKB supports research on efficient, general inference methods and optimized task-specific methods.

HPKB is a timely impetus for knowledge-based technology, though some may think it overdue. Some of the tenets of HPKB were voiced in 1987 by Lenat and Feigenbaum and some have been around for longer. Lenat's Cyc project has also contributed much to our understanding of large knowledge bases and ontologies. Now, thirteen years into the Cyc project and more than a decade after Lenat and Feigenbaum's paper, there seems to be consensus on the following points:

The first and most intellectually taxing task when building a large knowledge base is to design an ontology. If you get it wrong, you can expect ongoing trouble organizing the knowledge you acquire in a natural way. Whenever two or more systems are built for related tasks (e.g., medical expert systems, planning, modeling of physical processes, scheduling and logistics, natural language understanding), the architects of the systems realize, often too late, that someone else has already done, or is in the process of doing, the hard ontological work. HPKB challenges the research community to share, merge, and collectively develop large ontologies for significant military problems. However, an ontology alone is not sufficient. Axioms are required to give meaning to the terms in an ontology. Without them, users of the ontology may interpret the terms differently.

Paul Cohen

Most knowledge-based systems have no common sense so cannot be trusted. Suppose you have a knowledge-based system for scheduling resources such as heavy-lift helicopters, and none of its knowledge concerns noncombatant evacuation operations. Now suppose you have to evacuate a lot of people. Lacking common sense, your system is literally useless. With a little common sense, it could not only support human planning but might be superior to it, because it could think outside the box and consider using the helicopters in an unconventional way. Common sense is needed to recognize and exploit opportunities, as well as to avoid foolish mistakes.

Very often one will accept an answer that is roughly correct, especially when the alternatives are no answer at all or a very specific but wrong answer. This is Lenat and Feigenbaum's breadth hypothesis: "Intelligent performance often requires the problem solver to fall back on increasingly general knowledge, and/or to analogize to specific knowledge from far-flung domains." We must therefore augment high-power knowledge-based systems, which give specific and precise answers, with weaker but adequate knowledge and inference. The inference methods may not all be sound and complete. Indeed, one might need a multitude of methods to implement what Polya called plausible inference. HPKB encompasses work on a variety of logical, probabilistic, and other inference methods.

It is one thing to recognize the need for common sense knowledge, another to integrate it seamlessly into knowledge-based systems. Lenat observes that ontologies often are missing a "middle level," the purpose of which is to connect very general ontological concepts such as "human" and "activity" with domain-specific concepts such as "the person who is responsible for navigating a B-52 bomber." Because HPKB is grounded in domain-specific tasks, the focus of much ontological engineering is this middle layer.

The Participants

The HPKB participants are organized into three groups: technology developers, integration teams, and challenge problem developers. Roughly speaking, the integration teams build systems with the new technologies to solve challenge problems. The integration teams are led by SAIC and Teknowledge. Each fields systems to solve challenge problems in an annual evaluation. University participants include Stanford University, Massachusetts Institute of Technology, Carnegie Mellon University, Northwestern University, University of Massachusetts, George Mason University and the University of Edinburgh. In addition, SRI International, the University of Southern California Information Sciences Institute, the Kestrel Institute, and TextWise, Inc. have developed important components. Information Extraction and Transport, Inc., with Pacific Sierra Research, developed and evaluated the Crisis Management challenge problem, while Alphatech, Inc. is responsible for the Battlespace challenge problem.

Challenge Problems

A programmatic innovation of HPKB is challenge problems. The *Crisis Management* challenge problem, developed by Information Extraction and Transport (IET), Inc. and Pacific Sierra Research (PSR) Inc., is designed to exercise broad, relatively shallow knowledge about international tensions. The Battlespace challenge problem, developed by Alphatech, Inc., has two parts, each designed to exercise relatively specific knowledge about activities in armed conflicts. *Movement Analysis* involves interpreting vehicle movements detected and tracked by idealized sensors. The *Workarounds* problem is concerned with finding military engineering solutions to traffic obstruction problems such as destroyed bridges and blocked tunnels.

Good challenge problems must satisfy several, often conflicting criteria. A challenge problem must be challenging: it must raise the bar for both technology and science. A problem that requires only technical ingenuity will not hold the attention of the technology developers, nor will it help the U.S. maintain its preeminence in science. Equally important, a challenge problem for a DARPA program must have clear significance to the Department of Defense. Challenge problems should serve for the duration of the program, becoming more challenging each year. This is preferable to designing new problems every year because the infrastructure to support challenge problems is expensive.

A challenge problem should require little or no access to military subject matter experts. It should not introduce a knowledge acquisition bottleneck that results in delays and low productivity from the technology developers. As much as possible, the problem should be solvable with accessible, open-source material. A challenge problem should exercise all (or most) of the contributions of the technology developers and it should exercise an integration of these technologies. A challenge problem should have unambiguous criteria for evaluating its solutions. These criteria need not be so objective that one can write algorithms to score performance (e.g., human judgment might be needed to assess scores) but they must be clear and they must be published early in the program. And although performance is very important, challenge problem must not value performance above all else, as this encourages "one-off" solutions and discourages researchers from spending time trying to understand why their technologies work well and poorly. A challenge problem should provide a steady stream of results, so progress can be assessed not only by technology developers, but also by DARPA management and involved members of the DoD community.

The HPKB challenge problems are designed to support new and on-going DARPA initiatives in intelligence analysis and battlespace information systems. Crisis management systems will assist strategic analysts by evaluating the political, economic and military courses of action available to nations engaged at various levels of conflict. Battlespace systems will support operations officers and intelligence analysts by inferring militarily significant targets and sites, reasoning about road network trafficability, and anticipating responses to military strikes.

Paul Cohen

Crisis Management Challenge Problem

The Crisis Management challenge problem is intended to drive the development of broad, relatively shallow common sense knowledge bases to facilitate intelligence analysis. The client program at DARPA for this problem is Project Genoa – Collaborative Crisis Understanding and Management. Genoa is intended to help analysts more rapidly understand emerging international crises, so as to preserve U.S. policy options. Proactive crisis management — before a situation has evolved into crisis that may engage the U.S. military — enables more effective responses than reactive management. Crisis management systems will assist strategic analysts by evaluating the political, economic and military courses of action available to nations engaged at various levels of conflict.

The challenge problem development team worked with Genoa representatives to identify areas for the application of HPKB technology. This took three or four months, but the Crisis Management challenge problem specification has remained fairly stable since its initial release in draft form in July 1997.

The first step in creating the challenge problem was to develop a *scenario* to provide context for intelligence analysis in time of crisis. To ensure that the problem should require development of real knowledge about the world, the scenario includes real national actors with a fictional yet plausible story line. The scenario, which takes place in the Persian Gulf, involves hostilities between Saudi Arabia and Iran that culminate in closing the Strait of Hormuz to international shipping.

Next, IET worked with experts at PSR to develop a description of the intelligence analysis process, which involves the following tasks:

- Information gathering -- "What happened?"
- Situation assessment -- "What does it mean?"
- Scenario development -- "What might happen next?"

Situation assessment (or interpretation) includes factors that pertain to the specific situation at hand, such as motives, intents, risks, rewards, and ramifications; and factors that make up a general context, or "strategic culture," for a state actor's behavior in international relations, such as capabilities, interests, policies, ideologies, alliances, and enmities. Scenario development, or speculative prediction, starts with the generation of plausible actions for each actor. Then options are evaluated with respect to the same factors as for situation assessment, and a likelihood rating is produced. The most plausible actions are reported back to policy makers.

These analytic tasks afford many opportunities for knowledge-based systems. One is to use knowledge bases to retain or multiply corporate expertise; another is to use knowledge and reasoning to "think outside the box," to generate analytical possibilities that a human analyst might overlook. The latter task requires extensive common sense knowledge, or "analyst's sense," about the domain, to rule out implausible options.

The Crisis Management challenge problem includes an informal specification for a prototype *crisis management assistant* to support analysts. The assistant is tested by asking questions. Some are simple requests for factual information, others require the assistant to interpret the actions of nations in the context of strategic culture. Actions are motivated by interests, balancing risks and rewards. They have impacts

and require capabilities. Interests drive the formation of alliances, the exercise of influence, and the generation of tensions among actors. These factors play out in a current and historical context. Crises can be represented as events or as larger episodes tracking the evolution of a conflict over time, from inception or trigger, through any escalation, to eventual resolution or stasis. The representations being developed in HPKB are intended to serve a crisis corporate memory, to help analysts discover historical precedents and analogies for actions. Much of the challenge problem specification is devoted to *sample questions* that are intended to drive the development of general models for reasoning about crisis events.

Sample questions are embedded in an analytical context. The question "What might happen next?" is instantiated as "What might happen following the Saudi air strikes?" as shown in Figure 1. Q51 is refined to Q83 in a way that is characteristic of the analytical process; that is, higher-level questions are refined into sets of lower-level questions that provide detail.

III. What of significance might happen following the Saudi air strikes?
B. Options evaluation
Evaluate the options available to Iran
Close the Strait of Hormuz to shipping
Evaluation: Probable

Motivation: Respond to Saudi air strikes and deter future strikes

Capability:
(Q51) Can Iran close the Strait of Hormuz to international shipping?
(Q83) Is Iran capable of firing upon tankers in the Strait of Hormuz? With what weapons?
Negative outcomes:
(Q53) What risks would Iran face in closing the Strait?

Figure 1. Sample questions pertaining to the responses to an event.

The challenge problem developers (IET with PSR) developed an answer key for sample questions, a fragment of which is shown in Figure 2. While simple factual questions (e.g., "What is the GNP of the U.S.?") have just one answer, questions like Q53 usually have several. The answer key actually lists five answers, two of which are shown in Figure 2. Each is accompanied by suitable explanations, including source material. The first source (Convention on the Law of the Sea) is electronic. IET maintains a Web site with links to pages that are expected to be useful in answering the questions. The second "source" is a fragment of a model developed by IET and published in the challenge problem specification. IET developed these fragments to indicate the kinds of reasoning they would be testing in the challenge problem.

Answer(s):
- Economic sanctions from {Saudi Arabia, GCC, US, UN}
  - The closure of the Strait of Hormuz would violate an international norm promoting freedom of the seas and would jeopardize the interests of many states.
  - In response, states might act unilaterally or jointly to impose economic sanctions on Iran to compel it to re-open the Strait.

- The UN Security Council might authorize economic sanctions against Iran.
- Limited military response from {Saudi Arabia, GCC, US, others}...

Source(s):

- The Convention on the Law of the Sea.
- (B5) States may act unilaterally or collectively to isolate and/or punish a group or state that violates international norms. Unilateral and collective action can involve a wide range of mechanisms, such as intelligence collection, military retaliation, economic sanction, and diplomatic censure / isolation.

Figure 2. Part of the answer key for question 53

For the challenge problem evaluation, held in June 1998, IET developed a way to generate test questions through parameterization. Test questions deviate from sample questions in specified, controlled ways, so the teams participating in the challenge problem know the space of questions from which test items will be selected. This space includes billions of questions so cannot covered by question-specific knowledge. The teams must rely on general knowledge to perform well in the evaluation. (Semantics provides practical constraints the number of reasonable instantiations of parameterized questions, as do on-line sources provided by IET.) To illustrate, Q53 is parameterized in Figure 3. Parameterized question 53, PQ53, actually covers eight of the roughly one hundred sample questions in the specification.

PQ53 [During/After <TimeInterval>,] what {risks, rewards} would <InternationalAgent> face in <InternationalActionType>?

<InternationalActionType> =
{[exposure of its] {supporting, sponsoring}
<InternationalAgentType> in <InternationalAgent2>,
successful terrorist attacks against <InternationalAgent2>'s
<EconomicSector>,
<InternationalActionType>(PQ51),
taking hostage citizens of <InternationalAgent2>,
attacking targets <SpatialRelationship> <InternationalAgent2> with <Force>}

<InternationalAgentType> =
{terrorist group, dissident group, political party, humanitarian organization}

Figure 3. A parameterized question suitable for generating sample questions and test questions.

Parameterized questions and associated class definitions are based on natural language, giving the integration teams responsibility for developing (potentially different) formal representations of the questions. This decision was taken at the request of the teams. An instance of a parameterized question, say PQ53, is mechanically generated, then the teams must create a formal representation and reason with it – without human intervention.

Battlespace Challenge Problems

The second challenge problem domain for HPKB is battlespace reasoning. *Battlespace* is an abstract notion that includes not only the physical geography of a conflict but also the plans, goals, and activities of all combatants prior to and during a battle and during the

activities leading up to the battle. Three battlespace programs within DARPA were identified as potential users of HPKB technologies: the Dynamic Multi-Information Fusion program, the Dynamic Database program, and the Joint Forces Air Component Commander program. Two battlespace challenge problems have been developed.

The Movement Analysis Challenge Problem

The Movement Analysis challenge problem concerns high-level analysis of idealized sensor data, particularly the airborne JSTARS' Moving Target Indicator radar. This Doppler radar can generate vast quantities of information – one reading per minute for each vehicle in motion within a 10,000 square mile area.[3] The Movement Analysis scenario involves an enemy mobilizing a full division of ground forces – roughly 200 military units and 2000 vehicles – to defend against a possible attack. A simulation of the vehicle movements of this division was developed, the output of which includes reports of the positions of all of the vehicles in the division at one minute intervals over a four-day period, for eighteen hours each day. These military vehicle movements were then interspersed with plausible civilian traffic, to add the problem of distinguishing military from non-military traffic. The Movement Analysis task is to monitor the movements of the enemy to detect and identify types of military sites and convoys.

Because HPKB is not concerned with signal processing, the inputs are not real JSTARS data but are instead generated by a simulator and preprocessed into vehicle "tracks." There is no uncertainty in vehicle location and no radar shadowing, and each vehicle is always accurately identified by a unique "bumper number". However, vehicle tracks do not precisely identify vehicle type, but instead report each vehicle as being either light-wheeled, heavy-wheeled, or tracked. Low-speed and stationary vehicles are not reported.

Vehicle track data are supplemented by small quantities of high-value intelligence data, including accurate identification of a few key enemy sites, "electronic intelligence" reports of locations and times at which an enemy radar is turned on, "communications intelligence" reports that summarize information obtained by monitoring enemy communications, and "human intelligence" reports that provide detailed information about the numbers and types of vehicles passing a given location. Other inputs include a detailed road network in electronic form, and an order of battle that describes the structure and composition of the enemy forces in the scenario region.

Given these inputs, Movement Analysis comprises the following tasks:

- Distinguish military from non-military traffic. Almost all military traffic travels in convoys, which makes this a fairly straightforward task except for very small convoys of two or three vehicles.

---

[3] These numbers and all information regarding MTI radar are approximate; actual figures are classified.

- Identify the sites between which military convoys travel, determine which of these sites are militarily significant, and determine the types of each militarily significant site. Site types include battle positions, command posts, support areas, air defense sites, artillery sites, and assembly/staging areas.
- Identify which units (or parts of units) in the enemy order of battle are participating in each military convoy.
- Determine the purpose of each convoy movement. Purposes include reconnaissance, movement of an entire unit towards a battle position, activities by command elements, and support activities.

- Infer the exact types of the vehicles that make up each convoy. About twenty types of military vehicles are distinguished in the enemy order of battle, all of which show up in the scenario data.

To help the technology base and the integration teams develop their systems, a portion of the simulation data was released in advance of the evaluation phase, accompanied by an answer key that supplied "model answers" for each of the inference tasks listed above.

Movement analysis is currently carried out manually by human intelligence analysts, who appear to rely on models of enemy behavior at several levels of abstraction. These include models of how different sites or convoys are structured for different purposes, and models of military systems such as logistics (supply and resupply). For example, in a logistics model one might find the following fragment: "Each echelon in a military organization is responsible for resupplying its subordinate echelons. Each echelon, from battalion on up, has a designated area for storing supplies. Supplies are provided by higher echelons and transshipped to lower echelons at these areas." Model fragments such as these are thought to constitute the knowledge of intelligence analysts and thus should be the content of HPKB movement analysis systems. Some such knowledge was elicited from military intelligence analysts during program-wide meetings. These same analysts also scripted the simulation scenario.

The Workarounds Challenge Problem

The Workarounds CP supports air campaign planning by the Joint Forces Air Component Commander (JFACC) and his or her staff. One task for the JFACC is to determine suitable targets for air strikes. Good targets allow one to achieve maximum military effect with minimum risk to friendly forces and minimum loss of life on all sides. Infrastructure often provides such targets: It can be sufficient to destroy supplies at a few key sites or critical nodes in a transportation network, such as bridges along supply routes. However, bridges and other targets can be repaired, and there is little point in destroying a bridge if an available fording site is nearby. If a plan requires an interruption in traffic of several days, and the bridge can be repaired in a few hours, then another target might be more suitable. Target selection, then, requires some reasoning about how an enemy may "work around" the damage to the target.

The task of the Workarounds challenge problem is to automatically assess how rapidly and by what method an enemy can reconstitute or bypass damage to a target, and thereby

to help air campaign planners rapidly choose effective targets. The focus of the Workarounds problem in the first year of HPKB is automatic *workarounds generation.*

The Workarounds task involves detailed representation of targets and the local terrain around the target, and detailed reasoning about actions the enemy can take to reconstitute or bypass this damage. Thus the inputs to Workarounds systems include:

- A description of a target (e.g. a bridge or a tunnel), the damage to it (e.g. one span of a bridge is dropped; the bridge and vicinity are mined), and key features of the local terrain (e.g. the slope and soil types of a terrain cross-section coincident with the road near the bridge, together with the maximum depth and speed of any river or stream the bridge crosses).
- A specific enemy unit or capability to be interdicted, such as a particular armored battalion, or supply trucks carrying ammunition.
- A time period over which that unit or capability is to be denied access to the targeted route. The presumption is that the enemy will try to repair the damage within this time period; a target is considered to be effective if there appears to be no way for the enemy to do this.
- A detailed description of the enemy resources in the area that could be used to repair the damage. For the most part, repairs to battle damage are carried out by Army engineers, so this description takes the form of a detailed engineering order of battle.

All inputs are provided in a formal representation language.

The workarounds generator is expected to provide three outputs. First, a *reconstitution schedule* giving the capacity of the damaged link as a function of time since the damage was inflicted. For example, the workarounds generator might conclude that the capacity of the link is zero for the first 48 hours, but thereafter a temporary bridge will be in place that can sustain a capacity of 170 vehicles per hour. Second, a *time line of engineering actions* that the enemy might carry out to implement the repair, the time these actions require, and temporal constraints among them. If there appears to be more than one viable repair strategy, a time line should be provided for each. Third, a *set of required assets*: For each time line of actions, a description of the engineering resources that are used to repair the damage, and pointers to the actions in the time line that employ these assets. The reconstitution schedule provides the minimal information required to evaluate the suitability of a given target. The time line of actions provides an explanation to justify the reconstitution schedule. The set of required assets is easily derived from the time line of actions, and can be used to suggest further targets for pre-emptive air strikes against the enemy to frustrate its repair efforts.

A training data set was provided to help CP developers build their systems. It supplied inputs and outputs for several sample problems, together with detailed descriptions of the calculations carried out to compute action durations, lists of simplifying assumptions made to facilitate these calculations, and pointers to text sources for information on

engineering resources and their use (mainly Army Field manuals available on the World-Wide Web).

Workarounds generation requires detailed knowledge about the capabilities of the enemy's engineering equipment and how it is typically used by enemy forces. For example, repairing damage to a bridge typically involves mobile bridging equipment such as armored vehicle-launched bridges (AVLBs), medium girder bridges, Bailey bridges, or float bridges such as ribbon bridges or M4T6 bridges, together with a range of earthmoving equipment such as bulldozers. Each kind of mobile bridge takes a characteristic amount of time to deploy, requires different kinds of bank preparation, and is "owned" by different echelons in the military hierarchy, all of which affect the time it takes to bring the bridge to a damage site and effect a repair. Because HPKB operates in an entirely unclassified environment, U.S. engineering resources and doctrine were employed throughout. Information from Army Field Manuals was supplemented by a series of program-wide meetings with an army combat engineer, who also helped construct sample problems and solutions.

Integrated Systems

The challenge problems are solved by integrated systems fielded by integration teams led by Teknowledge and SAIC. Teknowledge favors a centralized architecture that contains a large common sense ontology (Cyc); SAIC has a distributed architecture that relies upon sharing specialized domain ontologies and knowledge bases, including a large "upper level" ontology based on the merging of Cyc, Sensus and other KBs.

Teknowledge Integration

The Teknowledge integration team comprises Teknowledge, Cycorp and Kestrel Institute. Its focus is on semantic integration and the creation of massive amounts of knowledge.

Semantic integration

Three issues make software integration difficult. *Transport* issues concern mechanisms to get data from one process or machine to another. Solutions include sockets, RMI and CORBA. *Syntactic* issues concern how to convert number formats, "syntactic sugar," and the labels of data. The more challenging issues concern *semantic integration*: To integrate elements properly, one must understand the meaning of each. The database community has addressed this issue (Wiederhold, 1996); it is even more pressing in knowledge based systems.

The current state of practice in software integration consists largely of interfacing pairs of systems as needed. Pairwise integration of this kind does not scale up, unanticipated uses

are hard to cover later, and chains of integrated systems at best evolve into stovepipe systems. Each integration is only as general as it needs to be to solve the problem at hand.

Some success has been achieved in low level integration and reuse; for example, systems that share scientific subroutine libraries or graphics packages are often forced into similar representational choices for low level data. DARPA has invested in early efforts to create reuse libraries for integrating large systems at higher levels (Carrico, 1997). Some effort has gone into expressing a generic semantics of plans in an object oriented format (Pease & Carrico, 1997; Pease & Carrico, 1997:2), and applications of this generic semantics to domain specific tasks is promising (Pease & Albericci, 1998). The development of ontologies for integrating manufacturing planning applications (Tate, 1998) and workflow (Lee, 1996) is ongoing.

Another option for semantic integration is software mediation (Park et al, 1997). This can be seen as a variant on pairwise integration, but because integration is done by knowledge-based means, one has an explicit expression of the semantics of the conversion. This renders the effort more reusable. Researchers at Kestrel Institute have successfully defined formal specifications for data and used those theories to integrate formally specified software (Srinivas & Jullig, 1995). In addition, researchers at Cycorp have successfully applied Cyc to the integration of multiple databases.

The Teknowledge approach to integration is to share knowledge among applications and create new knowledge to support the challenge problems. Teknowledge is defining formal semantics for the inputs and outputs of each application and the information in the challenge problems.

Many concepts defy simple definitions. While there has been much success in defining the semantics of mathematical concepts, it is harder to be precise about the semantics of the concepts people use every day. These seem to acquire meaning through their associations with other concepts, through their use in situations and communication, and through their relations to instances. To give the concepts in our integrated system real meaning we must provide a rich set of associations. This requires an extremely large knowledge base. Cyc offers just such a knowledge base.

Cyc

Cyc (Lenat, 1995; Lenat & Guha, 1990) consists of an immense, multi-contextual knowledge base, an efficient inference engine, and associated tools and interfaces for acquiring, browsing, editing, and combining knowledge. Its premise is that knowledge-based software will be less brittle if and only if it has access to a foundation of basic common sense knowledge. This semantic substratum of terms, rules, and relations enables application programs to cope with unforeseen circumstances and situations.

The Cyc knowledge base (KB) represents millions of hand-crafted axioms entered during the thirteen years since Cyc's inception. Through careful policing and generalizing, there are now slightly fewer than one million axioms in the knowledge base, interrelating roughly 50,000 atomic terms. Fewer than two percent of these axioms represent simple facts about proper nouns of the sort one might find in an almanac. Most embody general consensus information about the concepts. For example, one axiom says one cannot perform volitional actions while sleeping; another says one cannot be in two places at once; another says you must be at the same place as a tool to use it; and so on. The KB spans human capabilities and limitations, including information on emotions, beliefs, expectations, dreads, and goals; common everyday objects, processes and situations; and the physical universe, including such phenomena as time, space, causality, and motion.

The Cyc inference engine comprises an epistemological and a heuristic level. The epistemological level is an expressive nth-order logical language with clean formal semantics. The heuristic level is a set of some three dozen special-purpose modules that each contain their own algorithms and data structures, and are able to recognize and handle some commonly-occurring sorts of inference. For example, one heuristic level module handles temporal reasoning efficiently by converting temporal relations into a before-and-after graph, and then doing graph-searching rather than theorem-proving to derive an answer. A truth maintenance system and an argumentation-based explanation and justification system are tightly integrated into the system, and are efficient enough to be in operation at all times. In addition to these inference engines, Cyc includes numerous browsers, editors, and consistency-checkers. A rich interface has been defined.

Crisis Management Integration

The Crisis Management challenge problem involves answering test questions presented in a structured grammar. The first step in answering a test question is to convert it to a form that Cyc can reason with, a declarative decision tree. When the tree is applied to the test question input, a Cyc query is generated and sent to Cyc.

Answering the challenge problem questions takes a great deal of knowledge. For the first year's challenge problem, alone, the Cycorp and Teknowledge team added some 8,000 concepts and 80,000 assertions to Cyc. To meet the needs of this challenge problem the team created significant amounts of new knowledge, some developed by collaborators and merged into Cyc, some added by automated processes.

The Teknowledge integrated system includes two natural language components. The START system was created by Boris Katz and his group at MIT (Katz, 1997). For each of the Crisis Management questions, Teknowledge has developed a template into which user-specified parameters can be inserted. START parses English queries for a few of the Crisis Management questions to fill in those templates. Each filled template is a legal Cyc query. TextWise Corporation has been developing natural language information retrieval software primarily for news articles (Liddy, 1995). Teknowledge intends to use the TextWise KNOWledge base Information Tools (KNOW-IT) to supply many instances to Cyc of facts discovered from news stories. The system can parse English

text and return a series of binary relations that express the content of the sentences. There are several dozen relation types and the constants that instantiate each relation are Wordnet synset mappings (Miller, 1993). Each of the concepts has been mapped to a Cyc expression and a portion of Wordnet has been mapped to Cyc. For those synsets not in Cyc, the Wordnet hyponym links are traversed until a mapped Cyc term is found.

Battlespace integration

Movement Analysis

Several movement analysis systems were to be integrated, and much preliminary integration work was done. Ultimately the time pressure of the challenge problem evaluation precluded a full integration. The MIT and UMass movement analysis systems are described briefly here; the SMI and SRI systems are described in the section on SAIC Integration.

The MIT MAITA system provides tools for constructing and controlling networks of distributed monitoring processes. These tools provide access to large KBs of monitoring methods, organized around the hierarchies of tasks performed, knowledge used, contexts of application, alerting utility models, and other dimensions. Individual monitoring processes may also make use of KBs representing commonsense or expert knowledge in conducting their reasoning or reporting their findings. MIT built monitoring processes for sites and convoys with these tools.

The UMass group tried to identify convoys and sites with very simple rules. Rules were developed for three site types (battle positions, command posts, and assembly/staging areas). The convoy detector simply looked for vehicles traveling at fixed distances from one another. Initially, UMass was going to recognize convoys from their dynamics, in which the distances between vehicles fluctuate in a characteristic way, but in the simulated data the distances between vehicles remained fixed. UMass also intended to detect sites by the dynamics of vehicle movements between them, but no significant dynamical patterns could be found in the movement data.

Workarounds

Teknowledge developed two workarounds integrations, one an internal Teknowledge system, the other from AIAI at the University of Edinburgh.

Teknowledge developed a planning tool based on Cyc, essentially a wrapper around Cyc's existing KB and inference engine. A plan is a proof that there is a path from the final goal to the initial situation through a partially ordered set of actions. The rules in the KB driving the planner are rules about action preconditions and about which actions can bring about a certain state of affairs. There is no explicit temporal reasoning; the partial order of temporal precedence between actions is established on the basis of the rules about preconditions and effects.

Paul Cohen

The planner is a new kind of inference engine, performing its own search but in a much smaller search space. However, each step in the search involves interaction with the existing inference engine, by hypothesizing actions and microtheories and by doing `asks` and `asserts` in these microtheories. This hypothesizing and asserting on the fly in effect amounts to dynamically updating the KB in the course of inference; this is a new capability for the Cyc inference engine.

Consistent with the goals of HPKB the Teknowledge workaround planner re-used Cyc's knowledge, although it was not knowledge specific to workarounds. In fact, Cyc had never been the basis of a planner before, so even stating things in terms of an action's preconditions was new. What Cyc provided, however, was a rich basis on which to build workarounds knowledge. For example, the Teknowledge team needed to write only one rule to state "to use something as a device you must have control over that device," and this rule covered the cases of using an M88 to clear rubble, a mine plow to breach a minefield, a bulldozer to cut into a bank or narrow the gap, and so on. The reason one rule can cover so many cases is because clearing rubble, demining an area, narrowing a gap and cutting into a bank are all specializations of IntrinsicStateChangeEvent, an extant part of the Cyc ontology.

The AIAI workaround planner was also implemented in Cyc and took data from Teknowledge's Fire&ISE-to-MELD translator as its input. The central idea was to use the script-like structure of workaround plans to guide the reasoning process. For this reason a Hierarchical Task Network approach was taken. A planning-specific ontology was defined within the larger Cyc ontology and planning rules only referenced concepts within this more constrained context. The planning application was essentially embedded in Cyc.

Cyc had to be extended to represent composite actions that have several alternative decompositions and complex preconditions/effects. Although it is not a "commonsense" approach, AIAI decided to explore hierarchical task network planning as it appeared suitable for the workarounds domain. It was possible to represent actions, their conditions and effects, the plan node network, and plan resources in a relational style. The structure of a plan was implicitly represented in the proof that the corresponding composite action was a relation between particular sets of conditions and effects. Once proved, action relations are retained by Cyc and are potentially reusable. An advantage of implementing the AIAI planner in Cyc was the ability to remove brittleness from the planner input knowledge format; for instance, it was not necessary to account for all the possible permutations of argument order in predicates such as bordersOn and between.

SAIC Integrated System

SAIC built an HPKB Integrated Knowledge Environment (HIKE) to support both Crisis Management and Battlespace challenge problems. The architecture of HIKE for Crisis Management is shown in Figure SAIC-1. For Battlespace, the architecture is similar in that it is distributed and relies on the Open Knowledge Base Connectivity (OKBC) protocol, but of course the components integrated by the Battlespace architecture are different. HIKE's goals are to address the distributed communications and interoperability requirements among the HPKB technology components – knowledge servers, knowledge acquisition tools, question and answering systems, problem solvers,

monitoring processes, etc. – and to provide a graphical user interface tailored to the end users of the HPKB environment.

HIKE provides a distributed computing infrastructure that addresses two types of communications needs: First, input and output data transportation and software connectivities. These include connections between the HIKE server and technology components, connections between components, and connections between servers. HIKE encapsulates information content and data transportation through Java Objects, HyperText Transfer Protocol (HTTP), remote method invocation (Java RMI), database access (JDBC) and more. Second, HIKE provides for knowledge contents assertion and distribution, and query requests to knowledge services.

The OKBC protocol proved essential. SRI used it to interface the theorem prover SNARK to an OKBC server storing the CIA World Fact Book KB. Since the this knowledge base is large, SRI did not want to incorporate it into SNARK but instead used the procedural attachment feature of SNARK to look up facts that were available only in the World Fact Book. MIT's START system used OKBC to connect to SRI's Ocelot/SNARK OKBC server. This connection will eventually give users the ability to pose questions in English, which are then transformed to a formal representation by START, and shipped to SNARK using OKBC; the result is returned using OKBC. The Information Sciences Institute (ISI) built an OKBC server for their LOOM system for George Mason University. SAIC built a front end to the OKBC server for LOOM which was extensively used by the members of the battlespace challenge problem team.

With OKBC and other methods, the HIKE infrastructure permits the integration of new technology components (either clients or servers) in the integrated end-to-end HPKB system without introducing major changes, provided that the new components adhere to the specified protocols.

Crisis Management

The SAIC Crisis Management architecture is focused around a central OKBC bus as shown in Figure saic-1. The technology components provide user interfaces, question answering, and knowledge services. Some components have overlapping roles. For example, MIT's START system serves both as a user interface and a question answering component. Similarly, CMU's WebKB supports both question answering and knowledge services.

HIKE provides a form-based GUI with which users can construct queries with pull-down menus. Query construction templates correspond to the templates defined in the Crisis Management challenge problem specification. Questions also can be entered in natural language. START and the TextWise component accept natural language queries and then attempt to answer the questions. To answer questions that involve more complex types of reasoning, START generates a formal representation of the query and passes it to one of the theorem provers.

Paul Cohen

The Stanford Knowledge System Laboratory's (KSL) Ontolingua, SRI International's Graphical Knowledge Base (GKB) editor, WebKB, and TextWise provide the knowledge service components. The GKB editor is a graphical tool for browsing and editing large knowledge bases, used primarily for manual knowledge acquisition. WebKB supports semi-automatic knowledge acquisition. Given some training data and an ontology as input, a web spider searches in a directed manner and populates instances of classes and relations defined in the ontology. Probabilistic rules are also extracted. TextWise extracts information from text and newswire feeds, converting them into Knowledge Interchange Format (KIF) triples, which are then loaded into Ontolingua. Ontolingua is SAIC's central knowledge server and information repository for the Crisis Management challenge problem. Ontolingua supports KIF as well as Compositional Modeling Language (CML). Flow models developed by Northwestern University (NWU) answer Challenge Problem questions related to world oil transportation networks and reside within Ontolingua. Stanford University's System for Probabilistic Object Oriented Knowledge (SPOOK) provides a language for class frames to be annotated with probabilistic information, representing uncertainty about the properties of instances in that class. SPOOK is capable of reasoning with the probabilistic information based on Bayesian networks.

Question-answering is implemented in several ways. SRI International's SNARK and Stanford University's Abstract Theorem Prover (ATP) are first order theorem provers. WebKB answers questions based on the information it gathers. Question answering is also accomplished by START and TextWise taking a query in English as input and using information retrieval to extract the answers from text based sources (such as the web, newswire feeds, etc.).

The guiding philosophy during KB development for Crisis Management was to reuse knowledge whenever it made sense. The SAIC team reused three KBs: The HPKB upper level ontology developed by Cycorp, and World Fact Book KB from the CIA, and the Units and Measures Ontology from Stanford. Reusing the upper level ontology required translation, comprehension, and reformulation. The ontology was released in MELD (a language used by Cycorp) and was not directly readable by the SAIC system. In conjunction with Stanford, SRI developed a translator to load the upper level ontology into any OKBC-compliant server. Once loaded into the Ocelot server, the GKB-Editor was used to comprehend the upper ontology. The graphical nature of the GKB-Editor illuminated the inter-relationships between classes and predicates of the upper level ontology. As the upper level ontology represents functional relationships as predicates, but SNARK reasons efficiently with functions, it was necessary to reformulate the ontology to use functions whenever a functional relationship existed.

Paul Cohen

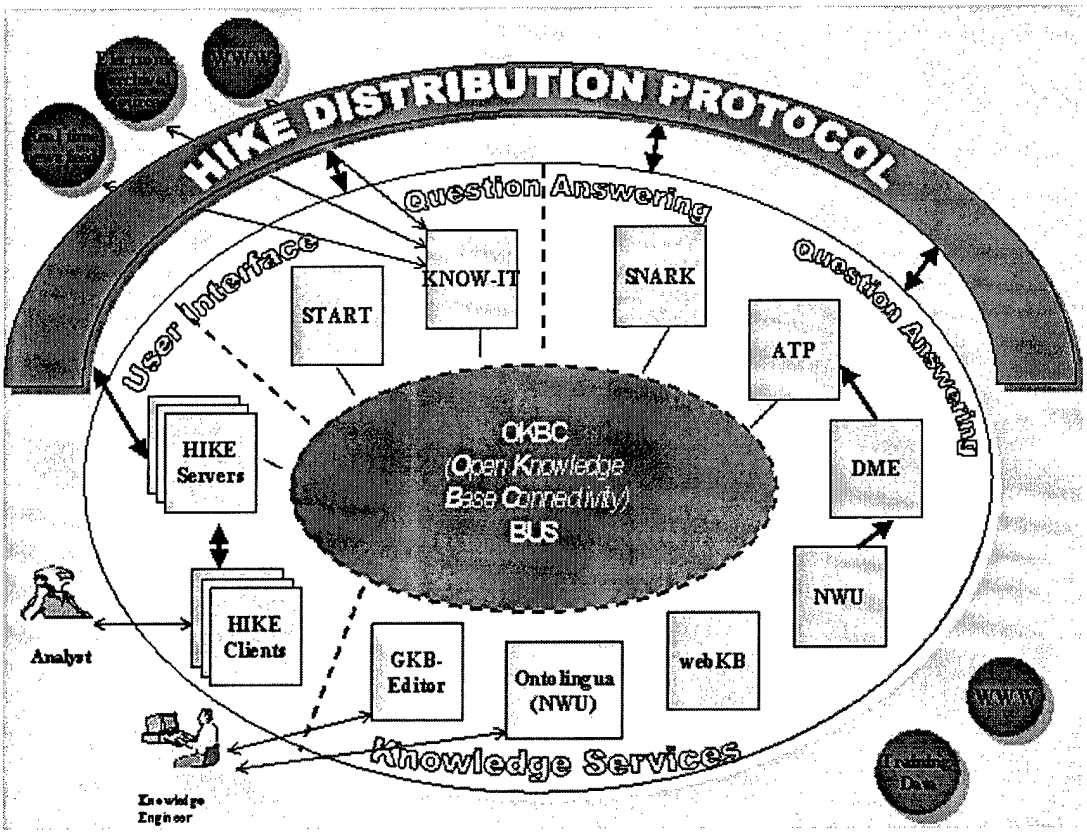


Figure SAIC-1. SAIC Crisis Management Challenge Problem Architecture

Battlespace Understanding

The distributed HIKE infrastructure is well suited to support an integrated Battlespace challenge problem as it was originally designed: a single information system for movement analysis, trafficability and workarounds reasoning. However, the trafficability problem (establishing routes for various kinds of vehicles given their characteristics) was dropped and the integration of the other problems was delayed. The components that solved these problems are described briefly, below.

Movement Analysis Integration

The Movement Analysis problem is solved by MIT's Monitoring, Analysis and Interpretation Tools Arsenal (MAITA), Stanford University's Section on Medical Informatics' (SMI) problem solving methods and SRI International's Bayesian Networks. The MIT effort was described briefly in the section on Teknowledge Integration. Here we focus on the SMI and SRI movement analysis systems.

Paul Cohen

For scalability, SMI adopted a three-layered approach to the challenge problem. The first layer consisted primarily of simple, context-free data processing that attempted to find important preliminary abstractions in the data set. The most important of these were *traffic centers* (locations which were either the start or stopping points for a significant number of vehicles) and *convoy segments* (a number of vehicles which depart from the same traffic center at roughly the same time, going in roughly the same direction). Spotting these abstractions required setting a number of parameters (e.g. how big is a traffic center?). Once trained, these first layer algorithms are linear in the size of the data set and enabled SMI to use knowledge-intensive techniques on the resulting (much smaller) set of data abstractions.

The second layer was a repair layer, which used knowledge of typical convoy behaviors and locations on the battlespace in order to construct a "map" of militarily significant traffic and traffic centers. The end result was a network of traffic connected by traffic. Three main tasks remain: classify the traffic centers, figure out what the convoys are doing, and identify which units are involved. SMI iteratively answered these questions by using repeated layers of heuristic classification and constraint satisfaction. The heuristic classification components operated independently of the network, using known (and deduced) facts about single convoys or traffic centers. Consider the following rule for trying to identify a main supply brigade (MSB) site (paraphrased into English, with abstractions in boldface)

```
If we have a current site which is unclassified
       and it's in the Division support area,
       and the traffic is high enough
       and the traffic is balanced
       and the site is persistent with no major deployments emanating from it
then it's probably an MSB
```

SMI used similar rules for the constraint satisfaction component of its system, allowing information to propagate through the network in a manner similar to Waltz's well-known constraint-satisfaction algorithm for edge labeling.<Waltz, 1975>

The goal of the SRI group was to induce a knowledge base to characterize and identify types of sites such as command posts, battle positions and so on. Their approach was to induce a Bayesian classifier, and use a generative model approach, producing a Bayesian network that could serve as a knowledge base. This required transforming raw vehicle tracks into features (e.g., the frequency of certain vehicles at sites, number of stops, etc.) which could be used to predict sites. Thus it was also necessary to have hypothetical sites to test. SRI relied on SMI to provide hypothetical sites and they also used some of the features SMI computed. As a classifier, SRI used TAN (Tree Augmented NaiveBayes; Friedman,Geiger, and Goldszmidt, 1997).

Workarounds Integration

The SAIC team integrated two approaches to workarounds generation, one developed by USC/ISI, the other by George Mason University (GMU).

ISI developed Course of Action Generation problem solvers to create alternative solutions to workarounds problems. In fact, two alternative Course of Action Generation problem solvers were developed. One is a novel planner whose knowledge base is represented in the ontologies, including its operators, state descriptions, and problem-specific information. It uses a novel partial match capability developed in Loom (MacGregor, 1991).The other is based on a state-of-the-art planner (Veloso et al., 1995). Each solution lists several engineering actions for that workaround (e.g., deslope the banks of the river, install a temporary bridge), includes information about there sources used (e.g. what kind of earthmoving equipment or bridge is used), and asserts temporal constraints among the individual actions to indicate which can be executed in parallel. A Temporal Estimation/Assessment problem solver evaluates each of the alternatives and selects one as the most likely choice for an enemy workaround. This problem solver was developed in EXPECT (Gil, 1994; Swartout and Gil, 1995).

Several general battlespace ontologies (e.g., military units, vehicles),anchored on the HPKB upper ontology, were used and augmented with ontologies needed to reason about workarounds (e.g., engineering equipment). Besides these ontologies, the knowledge bases used included a number of problem-solving methods to represent knowledge about how to solve the task. Both ontologies and problem-solving knowledge were used by two main problem solvers.

EXPECT's knowledge acquisition tools were used throughout the evaluation to detect missing knowledge. EXPECT uses problem-solving knowledge and ontologies to analyze which information is needed to solve the task. This capability allows EXPECT to alert a user when there is missing knowledge about a problem (e.g., unspecified bridge lengths) or a situation. It also helps debug and refine ontologies by detecting missing axioms and overgeneral definitions.

GMU developed the Disciple98 system. Disciple is an apprenticeship multistrategy learning system that learns from examples, from explanations, and by analogy, and can be taught by an expert how to perform domain-specific tasks through examples and explanations in a way that resembles how experts teach apprentices (Tecuci, 98). For the workarounds domain, Disciple was extended into a baseline integrated system that creates an ontology by acquiring concepts from a domain expert or importing them (through OKBC) from shared ontologies. It learns task decomposition rules from a domain expert and uses this knowledge to solve workarounds problems through hierarchical nonlinear planning.

First, with Disciple's ontology building tools, a domain expert assisted by a knowledge engineer built the object ontology from several sources, including expert's manuals, Alphatech's FIRE&ISE document and ISI's LOOM ontology. Second, a task taxonomy was defined by refining the task taxonomy provided by Alphatech. This taxonomy indicates principled decompositions of generic workaround tasks into subtasks, but does not indicate the conditions under which such decompositions should be performed. Third, the examples of hierarchical workaround plans provided by Alphatech were used to teach

Disciple. Each such plan provided Disciple with specific examples of decompositions of tasks into subtasks, and the expert guided Disciple to "understand" why each task decomposition was appropriate in that particular situation. From these examples and the explanations of why they are appropriate in the given situations Disciple learned general task-decomposition rules. After a knowledge base consisting of an object ontology and task decomposition rules was built, the hierarchical non-linear planner of Disciple was used to automatically generate workaround plans for new workaround problems.

Evaluation

The SAIC and Teknowledge integrated systems for Crisis Management, Movement Analysis and Workarounds were tested in an extensive study in June, 1998. The study followed a two-phase, test-retest schedule. In the first phase, the systems were tested on problems similar to those used for system development, but in the second, the problems required significant modifications to the systems. Within each phase the systems were tested and re-tested on the same problems. The test at the beginning of each phase established a baseline level of performance while the test at the end measured improvement during the phase.

We claim that HPKB technology facilitates rapid modification of knowledge-based systems. This claim was tested in both phases of the experiment, because each phase allows time to improve performance on test problems. Phase 2 provides a more stringent test: Only some of the phase 2 problems can be solved by the phase 1 systems, so the systems were expected to perform poorly in the test at the beginning of phase 2. The improvement in performance on these problems during phase 2 is a direct measure of how well HPKB technology facilitates knowledge capture, representation, merging, and modification.

Each challenge problem is evaluated by different metrics. The test items for Crisis Management were questions and the test was similar to an exam. Overall competence is a function of the number of questions answered correctly, but the Crisis Management systems are also expected to "show their work" and provide justifications (including sources) for their answers. Examples of questions, answers, and justifications for Crisis Management are shown in the section on the Crisis Management challenge problem, above.

Performance metrics for the Movement Analysis problem are related to recall and precision. The basic problem is to identify sites, vehicles, and purposes given vehicle track data, so performance is a function of how many of these entities are correctly identified and how many incorrect identifications are made. In general, identifications can be marked down on three dimensions: The identified entity may be more or less like the actual entity, the location of the identified entity can be displaced from the actual entity's true location, and the identification can be more or less timely.

The Workarounds problem involves generating workarounds to military actions such as bombing a bridge. Here, the criteria for successful performance include coverage (are all workarounds generated), appropriateness (are the generated workarounds appropriate given the action), specificity (how exactly is the workaround implemented), and accuracy of timing inferences (how long will each step in the workaround take to implement).

Performance evaluation, while essential, tells us relatively little about the HPKB integrated systems, still less the component technologies. We also want to know why the systems perform well or poorly. This involves credit assignment, as the systems comprise many technologies. We also want to gather evidence pertinent to several important, general claims. One such claim is that HPKB facilitates rapid construction of knowledge-based systems because ontologies and knowledge bases can be reused. The challenge problems by design involve broad, relatively shallow knowledge, in the case of Crisis Management, and deep, fairly specific knowledge in the Battlespace problems. It is unclear which kind of problem most favors the reuse claim, and why. We are developing analytical models of reuse. While the predictions of these models will not be directly tested in the first year's evaluation, we will gather data to calibrate these models for a later evaluation.

Results of the Challenge Problem Evaluation

Crisis Management

The evaluation of the SAIC and Teknowledge Crisis Management systems involved seven trials or batches of roughly 110 questions. Thus, more than 1500 answers were manually graded by the challenge problem developer, IET, and subject matter experts at PSR on criteria ranging from correctness to completeness of source material to the quality of the representation of the question. Each question in a batch was posed in English accompanied by the syntax of the corresponding parameterized question (see Fig. 3, above). The Crisis Management systems were supposed to translate these questions into an internal representation, MELD for the Teknowledge system and KIF for the SAIC system. The former translator was operational for all the trials, the latter was used to a limited extent on later trials.

The first trial involved testing the systems on the sample questions that had been available for several months for training. The remaining trials implemented the "test and retest with scenario modification" strategy discussed earlier. The first batch of test questions, TQA, was repeated four days later as a retest; it was designated TQA' for scoring purposes. The difference in scores between TQA and TQA' represents improvements to the systems. After solving the questions in TQA', the systems tackled a new set, TQB, designed to be "close to" TQA. The purpose of TQB was to check whether the improvements to the systems generalized to new questions. After a short break, a modification was introduced into the crisis management scenario and new fragments of knowledge about the scenario were released. Then the cycle repeated: A

new batch of questions, TQC, tested how well the systems coped with the scenario modification; then after four days the systems were retested on the same questions, TQC'; and on the same day a final batch, TQD, was released and answered.

Each question in a trial was scored according to several criteria, some "official" and others "optional." The four official criteria were the correctness of the answer, the quality of the explanation of the answer, the completeness and quality of cited sources, and the quality of the representation of the question. The optional criteria included lay intelligibility of explanations, novelty of assumptions, quality of the presentation of the explanation, how automatically the system produced a representation of the question, source novelty and reconciliation of multiple sources. Each question could garner a score between 0 and 3 on each criterion, and the criteria were themselves weighted. Some questions had multiple parts and the number of parts was a further weighting criterion. In retrospect it might have been clearer to assign each question a percentage of the points available, thus standardizing all scores, but in the data that follow, scores are on an open-ended scale. Subject matter experts were assisted with scoring the quality of knowledge representations when necessary.

A web-based form was developed for scoring, with clear instructions on how to assign scores. For example, on the "correct answer" criterion, the subject matter expert was instructed to award "Zero points if no top-level answer is provided and you cannot infer an intended answer; one point for a wrong answer without any convincing arguments, or most required answer elements; two points for a partially correct answer; three points for a correct answer addressing most required elements."

Considering the difficulty of the task, both systems performed remarkably well. Scores on the sample questions (SQ) were relatively high, which is not surprising because these questions had been available for training for several months (Figure CM-1). Nor is it surprising that scores on the very first batch of test questions (TQA) were not high. It is gratifying, however, to see how scores improve steadily between test and retest (TQA and TQA', TQC and TQC') and that these gains are *general*: The scores on TQA' and TQB, and TQC' and TQD were quite similar.

The scores designated "auto" in Figure CM-1 refer to questions that were translated automatically from English into a formal representation. The Teknowledge system translated all questions automatically, the SAIC system very few. Initially, the Teknowledge team did not manipulate the resulting representations, but in later batches they permitted themselves minor modifications. The effects of these can be seen in the differences between TQB and TQB-Auto, TQC and TQC-Auto, and TQD and TQD-Auto.

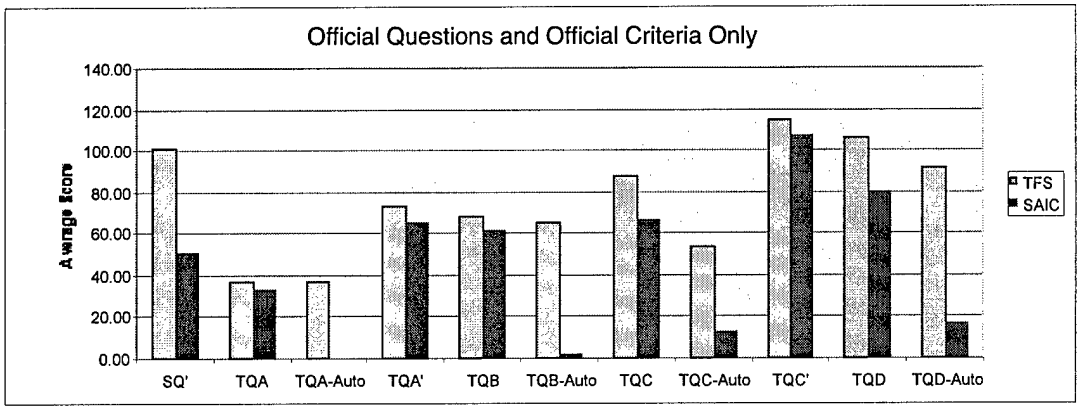


Figure CM-1. Scores on the seven batches of questions that made up the Crisis Management challenge problem evaluation. Some questions were automatically translated into a formal representation. The scores for these are denoted with the suffix "Auto."

While the scores of the Teknowledge and SAIC systems appear close in Figure CM-1, differences between the systems appear in other views of the data. Figure CM-2 shows the performance of the systems on all official questions plus a few optional questions. While these extra questions widen the gap between the systems, the real effect comes from adding "optional" components to the scores. Here, Teknowledge got credit for its automatic translation of questions, and also for the lay intelligibility of its explanations, the novelty of its assumptions and other criteria.

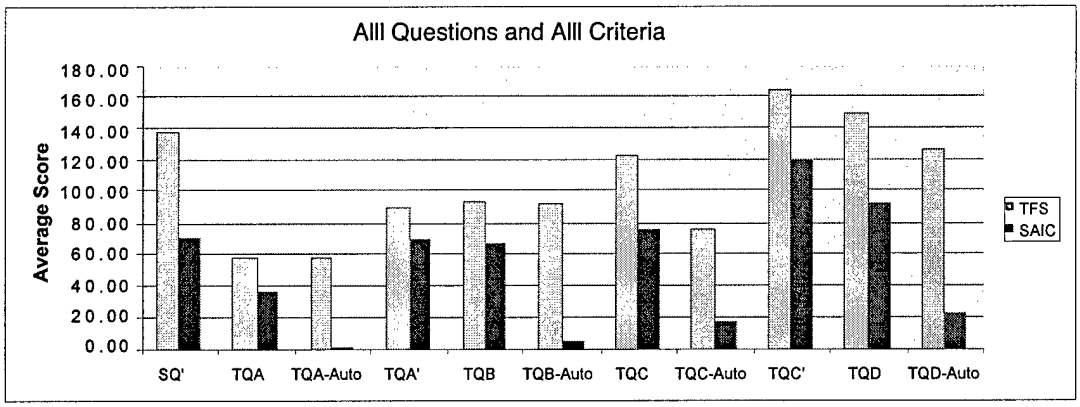


Figure CM-2. Scores on all questions (including optional ones) and both official and optional criteria.

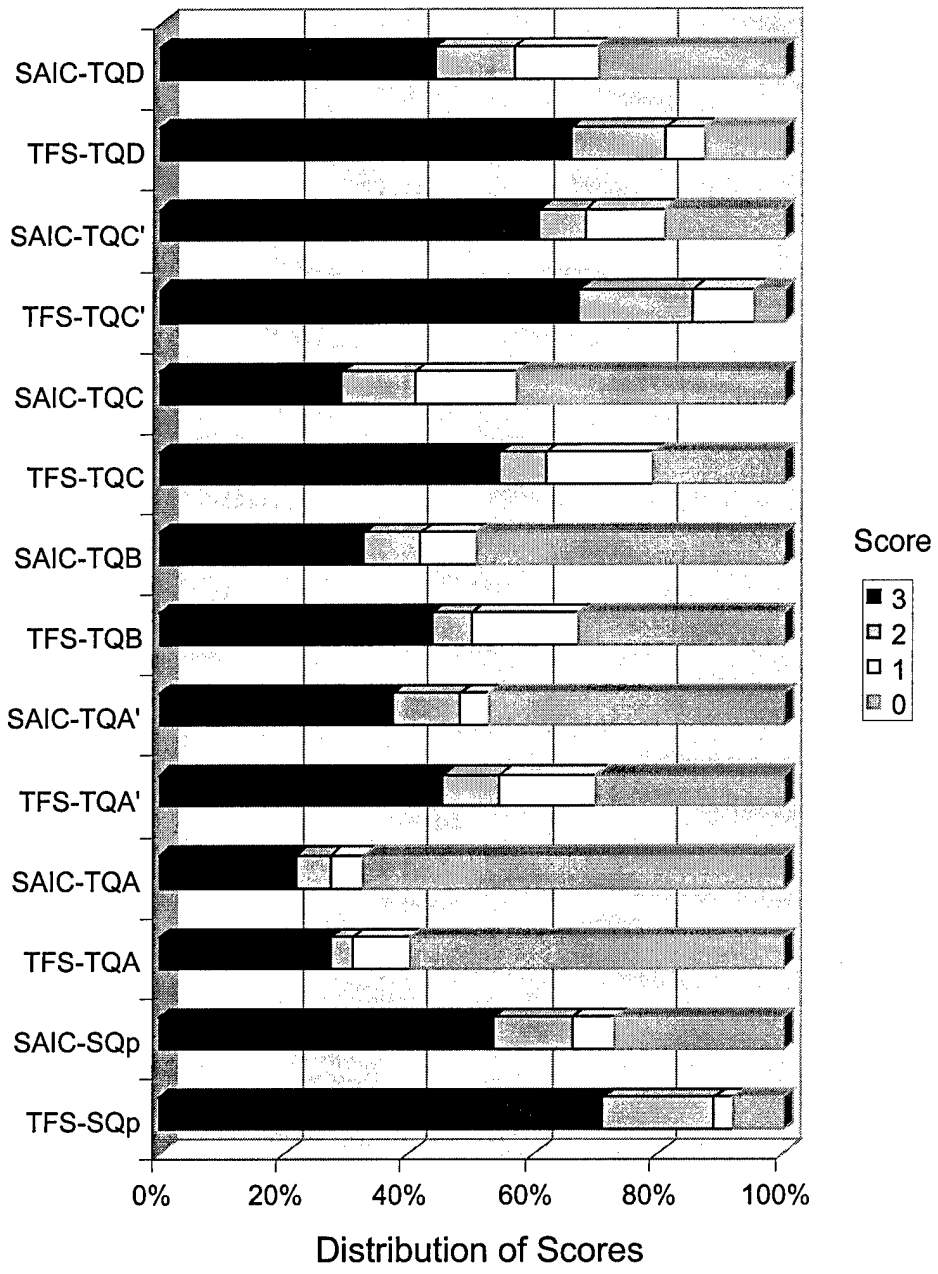


Figure CM-3. The distribution of correctness scores for the Crisis Management challenge problem. Correctness is one of the official scoring criteria, and takes values in {0,1,2,3}, with 3 being the highest score. This graph shows the proportion of the correctness scores that are 0, 1, 2, and 3. Looking at the lower bar, for example, roughly 70% of the correctness scores were 3 and roughly 10% were 1 or 0.

Recall that each question is given a score between zero and three on each of four official criteria. Figure CM-3 represents the distribution of the scores 0, 1, 2, 3 for the

correctness criterion over all questions for each batch of questions and each system. A score of 0 generally means the question was not answered. The Teknowledge system answered more questions than the SAIC system, and it had a larger proportion of perfect scores.

Interestingly, if one looks at the number of perfect scores as a fraction of the number of questions *answered*, the systems are not so different. One might argue that this comparison is invalid, that one's grade on a test depends on all the test items, not just those one answers. But suppose one is comparing students who have very different background and preparation; then it can be informative to compare them on both the number and quality of their answers. The Teknowledge/SAIC comparison is analogous to a comparison of students with very different backgrounds. Teknowledge used Cyc, a huge, mature knowledge base; whereas SAIC used Cyc's upper level ontology and a variety of disparate knowledge bases. The systems were not at the same "level of preparation" at the time of the experiment, so it is informative to ask how each system performed on the questions it answered. On the sample questions (SQ) Teknowledge got perfect scores on nearly 80% of the questions it answered while SAIC got perfect scores on nearly 70% of the questions it answered. But on TQA, TQA' and TQB, the systems are very similar, getting perfect scores on roughly 60%, 70% and 60% of the questions they answer. The gap widens to a roughly 10% spread in Teknowledge's favor on TQC, TQC' and TQD. Similarly, when one averages the scores of answered questions in each trial, the averages for the Teknowledge and SAIC systems are similar on all but the sample question trial. Even so, the Teknowledge system had both higher coverage (questions answered) and higher correctness on all trials.

If one looks at the minimum of the official component scores for each question – answer correctness, explanation adequacy, source adequacy and representation adequacy – the difference between the SAIC and Teknowledge systems is quite pronounced. Calculating the minimum component score is like finding something to complain about in each answer – the answer or the explanation, the sources or the question representation. It is the statistic of most interest to a potential user who would dismiss a system for failure on *any* of these criteria. Figure CM-4 shows that neither system did very well on this very stringent criterion, but the Teknowledge system got a perfect or good minimum score (3 or 2) roughly twice as often as the SAIC system in each trial.

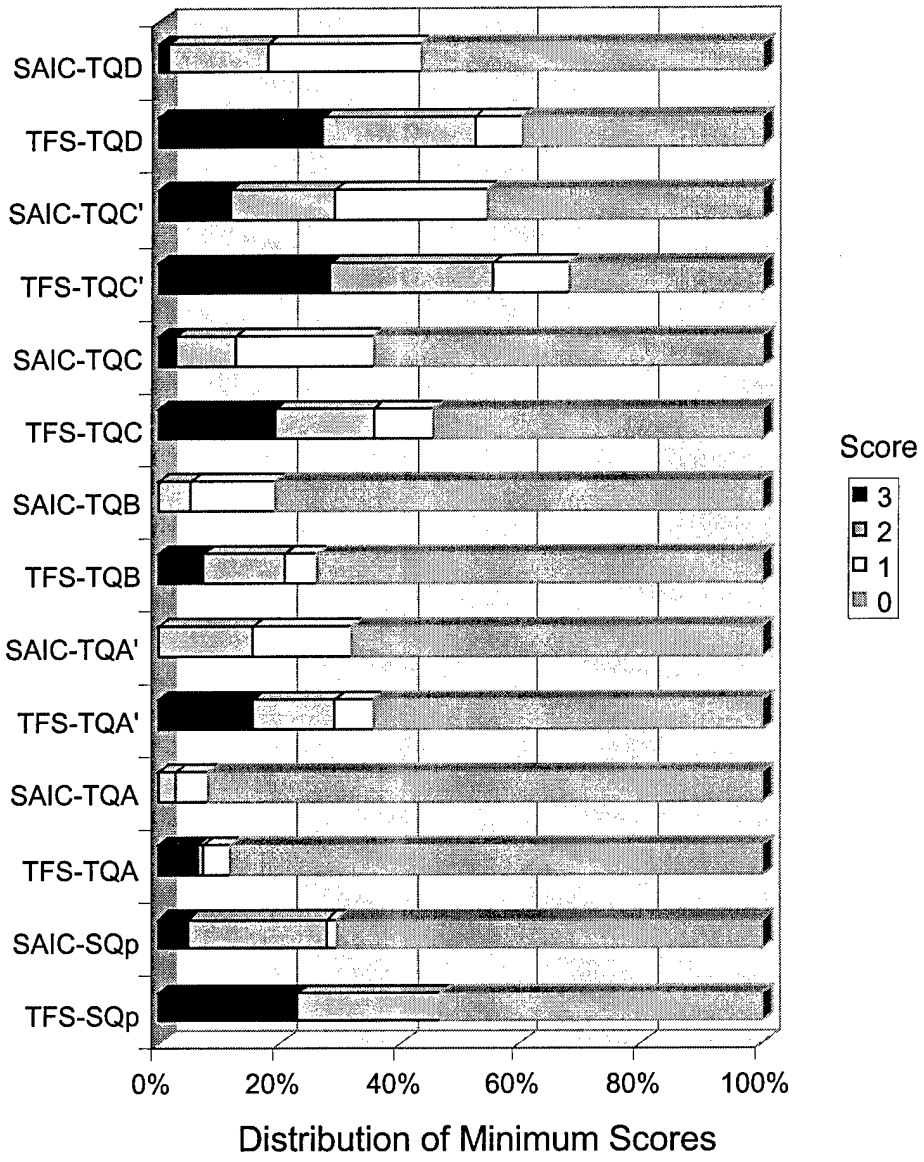


Figure CM-4. The distribution of minimum scores for the Crisis Management challenge problem Each question is awarded four component scores corresponding to the four official criteria. This graph shows the distribution of the *minimum* of these four components over questions. Looking at the lowest bar, for example, on 20% of the questions, the minimum of the four component scores was 3.

Movement Analysis

The task for movement analysis was to identify sites – command posts, artillery sites, battle positions and so on – given data about vehicle movements and some intelligence sources. Another component of the problem was to identify convoys. The data were

generated by simulation for an area of approximately 10,000 square kilometers over a period of approximately four simulated days. The scenario involved 194 military units arranged into five brigades, 1848 military vehicles, and 7666 civilian vehicles. Some units were co-located at sites, others stood alone. There were 726 convoys and nearly 1.8 million distinct reports of vehicle movements.

Four groups developed systems for all or part of the movement analysis problem, and SAIC and Teknowledge provided integration support. The groups were Stanford's Section on Medical Informatics (SMI), SRI, the University of Massachusetts (UMass), and MIT.

Each site identification was scored for its accuracy, and recall and precision scores were maintained for each site. Suppose an identification asserts at time t that a battalion command post exists at a location (x,y). To score this identification, we find all sites within a fixed radius of (x,y). Some site types are very similar to others; for example, all command posts have similar characteristics. So if one mistakes a battalion command post for, say, a division command post, then one should get partial credit for the identification. The identification is incorrect, but not as incorrect as, say, mistaking a command post for an artillery site. *Entity error* ranges from zero for completely correct identifications to one for hopelessly wrong identifications. Fractional entity errors provide partial credit for "near miss" identifications. If one of the sites within the radius of an identification matches the identification (e.g., a battalion command post) then the identification score is zero, but if none matches, then the score is the average entity error for the identification matched against all the sites in the radius. If no site exists within the radius of an identification then the identification is a *false positive.*

Recall and precision rates for sites are defined in terms of entity error. Let H be the number of sites identified with zero entity error, M be the number of sites identified with entity error less than one (near misses), and R be the number of sites identified with maximum entity error. Let T be the total number of sites, N be the total number of identifications, and F be the number of false positives. The following statistics describe the performance of the movement analysis systems:

zero entity error recall = H/T
non-one entity error recall = (H+M)/T
maximum entity error recall = (H+M+R)/T
zero entity error precision = H/N
non-one entity error precision = (H+M)/N
maximum entity error precision = (H+M+R)/N
false positive rate = F/N

Recall and precision scores for the groups are shown in Figure MA-1. The experiment design involved two phases with a test and retest within each phase. Additionally, the datasets for each test included either military traffic only or military plus civilian traffic. Had each group run their systems in each experimental condition there would have been 4 tests, 2 datasets with 2 versions of each (military only and military plus civilian), and 4

groups, or 64 conditions to score. All these conditions were not run. Delays were introduced by the process of reformatting data to make it compatible with a scoring program, so scores were not released in time for groups to modify their systems; one group devoted much time to scoring and did not participate in all trials; another group participated in no trials for reasons discussed at the end of this section.

The trials that were run are reported in Figure MA-1. Recall rates range from 40% to just over 60%, but these are for maximum entity error recall – the frequency of detecting a site when one exists, but not identifying it correctly. Rates for correct and near miss identification are lower, ranging from 10% to 15%, and are not shown in Figure MA-1. Of the participating research groups, MIT did not attempt to identify the types of sites, only whether sites are present, so their entity error is always maximum. The other groups did not attempt to identify all kinds of sites; for example, UMass tried to identify only battle positions, command posts and assembly/staging areas. Even so, each group was scored against all site types. Obviously the scores would be somewhat higher had the groups been scored against only the kinds of sites they intended to identify (e.g., recall rates for UMass ranged from 20% to 39% for the sites they tried to identify).

Precision rates are reported only for the SMI and UMass teams, as MIT did not try to identify the types of sites. UMass's precision was highest on their first trial; a small modification to their system boosted recall but at a significant cost to precision. SMI's precision hovered around 20% on all trials.

Scores were much higher for convoy detection. While scoring convoy identifications posed some interesting technical challenges, the results were plain enough: SMI, UMass and MIT detected 507 (70%), 616 (85%) and 465 (64%) of the 726 convoys, respectively. The differences between these figures do not indicate that one technology was superior to another because each group emphasized a slightly different aspect of the problem. For example, SMI didn't try to detect small convoys (fewer than four vehicles). In any case, the scores for convoy identifications are quite similar.

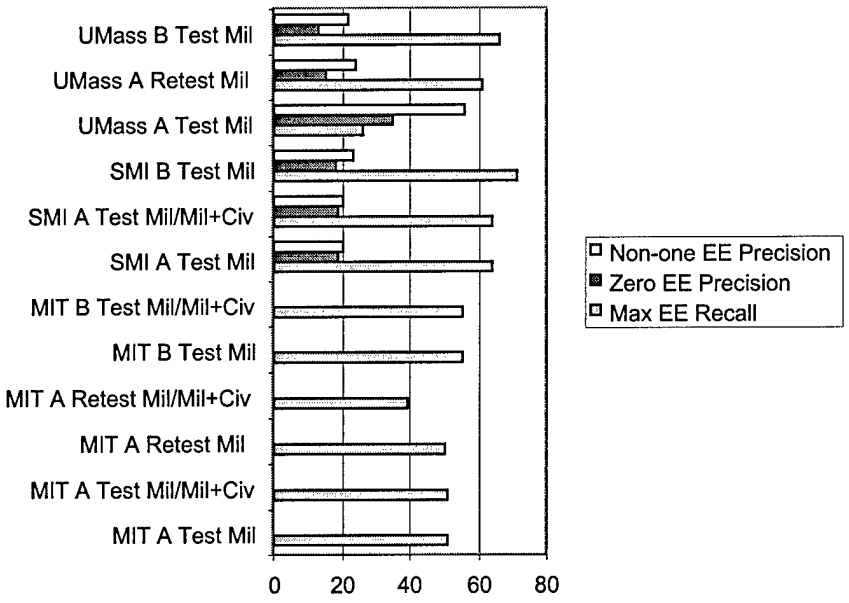


Figure MA-1. Scores for detecting and identifying sites in the Movement Analysis problem. Trials labeled "A" are from the first phase of the experiment, before the scenario modification. Trials labeled "B" are from after the scenario modification. "Mil" and "Mil+Civ" refer to datasets with military traffic only and with both military and civilian traffic, respectively.

In sum, none of the systems identified site types accurately, although all detected sites and convoys pretty well. The reason for these results is that sites can be detected by observing vehicle halts, just as convoys can be detected by observing clusters of vehicles. But it is difficult to identify site types without more information. It would be worth studying the types of information that humans require for the task.

The experience of SRI is instructive: The SRI system used features of movement data to infer site types, unfortunately, the data did not seem to support the task. There were insufficient quantities of training data to train classifiers (fewer than 40 instances of sites), but more importantly, the data did not have sufficient underlying structure – it was too random. The SRI group tried a variety of classifier technologies, including ID trees, nearest neighbor and c4.5, but classification accuracy remained in the low 30% range. This is comparable to the figures released by UMass on the performance of their system on three kinds of site. Even when UMass' recall figures are not diluted by sites they weren't looking for, they did not exceed 40%. SRI and UMass both performed extensive exploratory data analysis, looking for statistically significant relationships between features of movement data and site types, with little success, and the disclosed regularities were not strongly predictive. The reason for this is probably that the data were artificial. It is extraordinarily difficult to build simulators that generate data with the rich dependency structure of natural data.

Paul Cohen

Some simulated intelligence and radar information was released with the movement data. While none of the teams reported finding it useful, we believe this kind of data probably helps human analysts. One assumes humans perform the task better than the systems reported here, but as no human ever solved these movement analysis problems, one cannot tell whether the systems performed comparatively well or poorly. In any case, the systems took a valuable first step toward movement analysis, detecting most convoys and sites. Identifying the sites accurately will be a challenge for the future.

Workarounds

The workarounds problem was evaluated in much the same way as the other problems: The evaluation period was divided into two phases and within each phase the systems were given a test and a retest on the same problems. In the first phase the systems were tested on twenty problems and re-tested after a week on the same problems; in the second phase a modification was introduced into the scenario, the systems were tested on five problems, and after a week re-tested on the same five problems and five new ones.

Solutions were scored along five equally-weighted dimensions: Viability of enumerated workaround options, correctness of the overall time estimate for a workaround, correctness of solution steps provided for each viable option, correctness of temporal constraints among these steps, and appropriateness of engineering resources employed. Scores were assigned by comparing the systems' answers with those of human experts. Bonus points were awarded when, occasionally, systems gave better answers than the experts. These answers became the gold standard for scoring answers when the systems were re-tested.

Four systems were fielded by ISI, George Mason University, Teknowledge, and AIAI (Edinburgh). The results are shown in Figures WA-1 and WA-2. In each figure, the upper extreme of the vertical axis represents the maximum score a system could get by answering all the questions correctly (i.e., 200 points for the initial phase, 50 points for the first test in the modification phase, 100 points for the re-test in the modification phase). The bars represent the number of points scored, and the circles represent the number of points that could have been awarded given the number of questions that were actually answered. For example, in the initial phase, ISI answered all questions so could have been awarded 200 points on the test and 200 on the re-test; GMU covered only a portion of the domain and it could have been awarded a maximum of 90 and 100 points, respectively. The bars represent the number of points actually scored by each system..

How one views the performance of these systems depends on how one values correctness, coverage (the number of questions answered) and, more subtly, the prospects for scaling the systems to larger problem sets. An assistant should answer any question posed to it, but if the system is less than ideal should it answer more questions with some errors or fewer questions with fewer errors? Obviously the answer depends on the severity of errors and on the application, on the prospect for improving system coverage and for improving accuracy. What we might call "errors of specificity," in which an answer is less specific or complete than it should be, are not inconsistent with the

philosophy of HPKB, which expects systems to give partial – even common sense – answers when they lack specific knowledge.

Figures WA-1 and WA-2 show that ISI was the only group to attempt to solve all the workaround problems, although its answers were not all correct; whereas GMU solved fewer problems with higher overall correctness. AIAI solved fewer problems still, but quite correctly, and Teknowledge solved more problems than AIAI with more variable correctness. One can compute coverage and correctness scores as follows: Coverage is the number of questions attempted divided by the total number of questions in the experiment (55 in this case). Correctness is the total number of points awarded divided by the number of points that might have been awarded given the number of answers attempted. Figure WA-3 shows a plot of coverage against correctness for all the workaround systems. Points above and to the right of other points are superior, thus, the ISI and GMU systems are preferred to the other systems, but the ranking of these systems depends on how one values coverage and correctness.

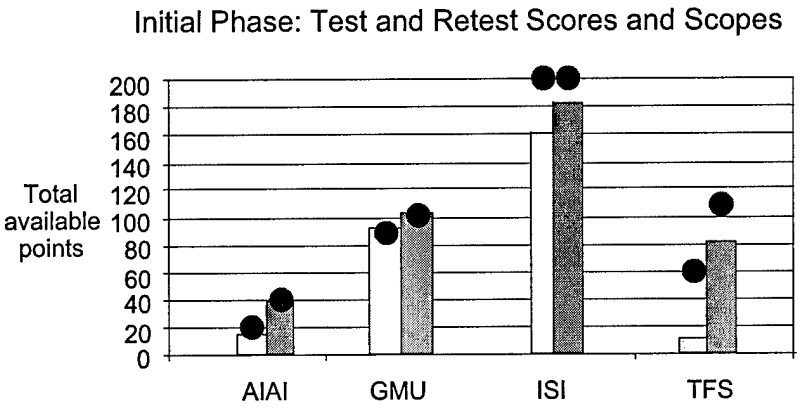


Figure WA-1. Results of the initial phase of the workaround evaluation. Lighter bars represent test scores, darker bars, retest scores. Circles represent the "scope" of the task attempted by each system, i.e., the best score that the system could have received given the number of questions it answered.

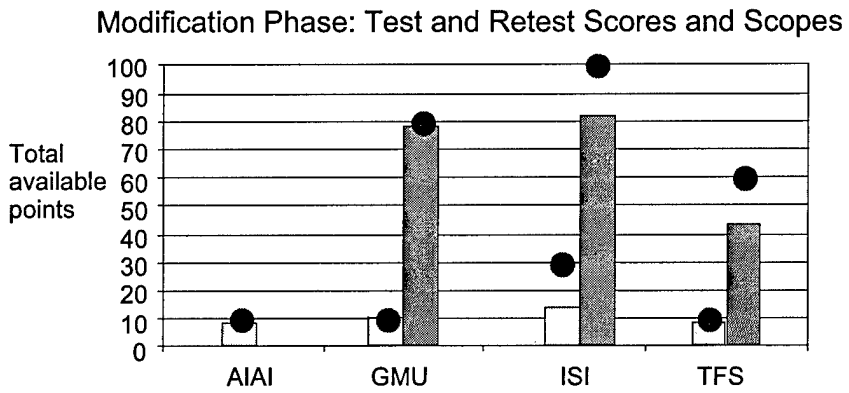


Figure WA-2. Results of the modification phase of the workaround evaluation. Lighter bars represent test scores, darker bars, retest scores. Circles represent the "scope" of the task attempted by each system, i.e., the best score that the system could have received given the number of questions it answered. Note that only five problems were released for the test, ten for the re-test, so the maximum available points for each are 50 and 100, respectively.

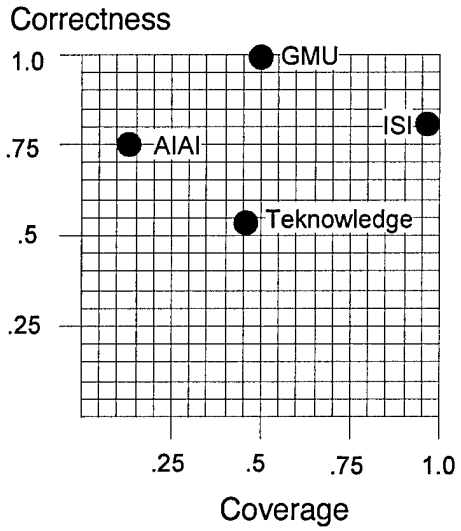


Figure WA-3. A plot of overall coverage against overall correctness for each system on the Workarounds challenge problem.

Two factors complicate comparisons between the systems. First, if one is allowed to select a subset of problems for one's system to solve, one might be expected to select problems on which one hopes the system will do well. Thus ISI's correctness scores are not strictly comparable to those of the other systems, because ISI did not select problems but solved them all. Said differently, what we call correctness is really "correctness in one's declared scope." It is not difficult to get a perfect correctness score if one selects just one problem; conversely, it is not easy to get a high correctness score if one solves all problems.

Second, Figure WA-3 shows that coverage can be increased by knowledge engineering, but at what cost to correctness? GMU believes correctness need not suffer with increased coverage, and cites the rapid growth of its KB. At the beginning of the evaluation period the coverage of the KB was about 40% (11841 binary predicates) and two weeks later the

coverage had grown significantly to roughly 80% of the domain (20324 binary predicates). In terms of coverage and correctness (Figure WA-3), GMU's coverage increased from 47.5% in the test phase to 80% in the modification phase with almost no decrement in correctness. On the other hand, ISI suggests that the complexity of the workarounds task increases significantly as the coverage is extended, to the point where a perfect score may be out of reach given the state of the art in AI technology. For example, in some cases the selection of resources requires combining plan generation with reasoning about temporal aspects of the (partially generated) plan. Moreover, interdependencies in the knowledge bases grow non-linearly with coverage, because different aspects of the domain interact and must be coordinated. For example, including mine-clearing operations changes the way one looks at how to ford a river. Such interactions in large knowledge bases may degrade the performance of a system when changes are made during a retest and modification phase.

The coverage vs. correctness debate should not cloud the accomplishments of the Workarounds systems. Both ISI and GMU were judged to perform the workarounds task at an expert level. All the systems were developed quite rapidly – indeed, GMU's KB doubled during the experimental period itself – and knowledge reuse was prevalent. These results take us some way toward the HPKB goal of very rapid development of powerful and flexible KB systems.

Evaluation Lessons Learned

The experiments reported here constitute one of the larger studies of AI technology. In addition to the results of these experiments, much was learned about how to conduct such studies. The primary lesson is that one should release the challenge problem specifications early. Many difficulties can be traced to delays in the release of the Battlespace specifications. Experiment procedures that involve multiple sites and technologies must be debugged in "dry run" experiments before the evaluation period begins. There must be agreement on the formats of input data, answers, and answer keys – a mundane requirement but one that caused delays in movement analysis evaluation.

A major contributor to the success of the Crisis Management problem was the parameterized question syntax, which gave all participants a concise representation of the kinds of problems they would solve during the evaluation. Similarly, evaluation metrics for Crisis Management were published relatively early. Initially, there was a strong desire for evaluation criteria to be objective enough for a machine to score performance. Relaxing this requirement had a very positive effect in Crisis Management. The criteria were objective in the sense that experts followed scoring guidelines, but we could ask the experts to assess the quality of a system's explanation – something no program can do. However, scoring roughly 1500 answers in Crisis Management took its toll on the experts and IET. It is worth considering whether equally informative results could be had at a lower cost.

Paul Cohen

The HPKB evaluations were set up as friendly competitions between the integration teams, SAIC and Teknowledge, each incorporating a subset of the technology in a unique architecture. However, some technology was used by *both* teams, and some was not tested in integrated systems but rather as standalone components. The purpose of the friendly competition – to evaluate merits of different technologies and architectures – was not fully realized. Given the cost of developing HPKB systems, we might re-evaluate the purported and actual benefits of competitions.

Conclusion

HPKB is an ambitious program, a high-risk, high-payoff venture beyond the edge of knowledge-based technology. In its first year HPKB suffered some setbacks, but these were informative and will help the program in future. More importantly, HPKB celebrated some extraordinary successes. It has long been a dream in Artificial Intelligence to ask a program a question in natural language about almost anything and receive a comprehensive, intelligible, well-informed answer. While we have not achieved this aim, we are getting close. In HPKB, questions were posed in natural language, questions on a hitherto impossible range of topics were answered, and the answers were supported by page after page of sources and justifications. Knowledge bases were constructed rapidly from many sources, re-use of knowledge was a significant factor, and semantic integration across knowledge bases started to become feasible. HPKB has integrated many technologies, cutting through traditional groupings in AI, to develop new ways of acquiring, organizing, sharing and reasoning with knowledge; and it has nourished a remarkable assault on perhaps the grandest of the AI grand challenges – to build intelligent programs that answer questions about everyday things: important things like international relations but also ordinary things like river banks.

Notes and References

Many HPKB resources and documents are found at http://www.teknowledge.com/HPKB. A web site devoted to the Year 1 HPKB Challenge Problem Conference can be found at http://www.teknowledge.com/HPKB/meetings/Year1meeting/. Definitive information on the Year 1 Crisis Management challenge problem, including updated specification, evaluation procedures, and evaluation results, is available at http://www.iet.com/Projects/HPKB/Y1Eval/. Earlier, draft information about both challenge problems is available at http://www.iet.com/Projects/HPKB/Combined-CPs.doc. Scoring procedures and data for the Movement Analysis problem is available at http://eksl-www.cs.umass.edu:80/research/hpkb/scores/. For additional sources, contact the authors.

References

Paul Cohen

Carrico, T., (1997) Object Model Working Group, Command and Control Schema, Revised Draft, Version 0.5.3. Unpublished report.

Cohen, Paul R. (1998). Dynamic Maps as Representations of Verbs. To appear in the Proceedings of the European Conference on Artificial Intelligence.

Friedman, Geiger, D., and Goldszmidt, M. Bayesian Network Classifiers, Machine Learning, 29, 131-163, 1997.

Gil, Y. "Knowledge refinement in a reflective architecture". In Proceedings of the Twelfth National Conference on Artificial Intelligence.1994.

Jones, E., (1998), "Formalized Intelligence Report Ensemble (FIRE) & Intelligence Stream Encoding (ISE)", Alphatech memo.

Katz, B. (1997), ``From Sentence Processing to Information Access on the World Wide Web," AAAI Spring Symposium on Natural Language Processing for the World Wide Web, Stanford University, Stanford CA.

Lee et al, (1996) The PIF Process Interchange and Framework Version 1.1, Jintae Lee (editor) Michael Grunninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost and other members of the PIF Working Group, published as MIT Center for Coordination Science, Working Paper #194, 1996. Available via http://soa.cba.hawaii.edu/pif/

Lenat, D. and Guha R., (1990) Building Large Knowledge Based Systems. Reading, Massachusetts: Addison Wesley.

Lenat, D. (1995) Artificial Intelligence. *Scientific American*, September.

Lenat, D. B. and Feigenbaum, E. A. 1987. On the Thresholds of Knowledge. Proceedings of the Tenth International Conference on Artificial Intelligence. Morgan Kaufmann. pp. 1173-1182.

Liddy, E.D., Paik, W. & McKenna, M. (1995). Development and Implementation of a Discourse Model for Newspaper Texts. In Proceedings of the AAAI Symposium on Empirical Methods in Discourse Interpretation and Generation. Stanford, CA.

MacGregor, R. "The evolving technology of classification-based knowledge representation systems". In Sowa, J., ed., "Principles of Semantic Networks: Explorations in the Representation of Knowledge". San Mateo, CA: Morgan Kaufmann. 1991.

Miller, G., Beckwith, R., Fellbaum, C, Gross, D., and Miller, K., (1993), Introduction to WordNet:An On-line Lexical Database. Available at http://www.cogsci.princeton.edu/~wn/papers/

Park, J., Gennari, J., & Musen, M., (1997). Mappings for Reuse in Knowledge-based Systems. Stanford Section on Medical Informatics technical report SMI-97-0697.

Pease, A., & Albericci, D., (1998). The Warplan, Teknowledge, Palo Alto, CA, March 13.

Pease, A. & Carrico, T. (1997). "The JTF ATD Core Plan Representation: Request for Comment". Armstrong Lab: AL/HR-TP-96-9631.

Pease, A. & Carrico, T. (1997). "The JTF ATD Core Plan Representation: A Progress Report", Proceedings of the AAAI 1997 Spring Symposium on Ontological Engineering.

Srinivas, Y. & Jullig, R., (1995) Specware(TM): Formal Support for Composing Software, Proceedings of the Conference on Mathematics of Program Construction, Kloster Irsee, Germany, July.

Swartout, B., and Gil, Y. "EXPECT: Explicit Representations for Flexible Acquisition". In Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop. Banff, Canada. 1995.

Paul Cohen

Tate, A. (1998), "Roots of SPAR - Shared Planning and Activity Representation", to appear in Knowledge Engineering Review, Special Issue on Ontologies, Cambridge University Press.

Tecuci G. (1998). Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies, Academic Press.

Veloso, M., Carbonell J., Perez, A, Borrajo, D., Fink, E., and Blythe, J. "Integrating Planning and Learning: The PRODIGY Architecture". Journal of Experimental and Theoretical AI, Vol 7, 1995.

Wiederhold, G. (ed.) (1996): Intelligent Integration of Information; Kluwer Academic Publishers, Boston MA, July 1996.